

AD 694090

MATHEMATICAL MODELS OF INFORMATION SYSTEMS

Professor Harvey L. Garner
University of Michigan

This document has been approved
for public release and sale; its
distribution is unlimited.

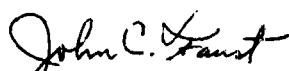
FOREWORD

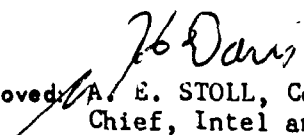
This technical report was submitted by the System Engineering Laboratory, Department of Electrical Engineering, The University of Michigan, Ann Arbor, Michigan 48104 under contract AF30(602)-3546, project 5581, task 558109. The period covered by this report extended from October 1967 to October 1968.

The project engineer was John C. Faust, Rome Air Development Center, EMIIS, Griffiss Air Force Base, N.Y.

This technical report has been reviewed by the Information Office and is releasable to the Clearing house for Federal Scientific and Technical Information.

This technical report has been reviewed and is approved.


Approved: JOHN C. FAUST
Project Engineer


Approved: A. E. STOLL, Colonel, USAF
Chief, Intel and Info Processing Division

FOR THE COMMANDER:


IRVING J. GABELMAN
Chief, Advanced Studies Group

ABSTRACT

This report summarizes research in the development of mathematical models of information processing systems conducted at The University of Michigan Systems Engineering Laboratory during the period from October, 1967 to October, 1968 under Rome Air Development Center sponsorship. Particular attention is given to a new approach to automata theory, the use of multiple index matrices in generalized automata theory, asymptotic decomposition of machines, recognizability of equation sets, algebraic isomorphism invariants for transition graphs, iterative network realization of sequential machines, optimum sequencing of jobs subject to deadlines, and the theory of formal languages and its impact on the design and implementation of programming languages.

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
ABSTRACT	iii
INTRODUCTION	1
1 A New Approach to Automata Theory by D. E. Muller	3
2 Use of Multiple Index Matrices in Generalized Automata Theory by D. E. Muller	48
3 Asymptotic Decomposability of Machines by G. R. Putzolu and D. E. Muller	64
4 The Recognizability of Equational Sets by C. D. Shepard	79
5 Algebraic Isomorphism Invariants for Transition Graphs by J. F. Meyer	83
6 Iterative Network Realization of Sequential Machines by J. R. Jump	105
7 Optimal Sequencing of Jobs Subject to Deadlines by E. L. Lawler and M. Moore	111
8 The Theory of Formal Languages and Its Impact on the Design and Implementation of Programming Languages by I. Y. Liu	127
REFERENCES	133

INTRODUCTION

This report is the final report in a series of yearly reports over a four-year period concerning mathematical models of information processing systems conducted in the Systems Engineering Laboratory under the direction of Professor H. L. Garner. Most of the theories relevant to computation relate to either automata theory or the theory of formal languages. This is true for most of the research discussed in this report. The exception is Section 7. In Section 7 the techniques common to operations research are used to investigate the problem of optimum sequencing.

In Section 1 Professor D. Muller outlines a new approach to automata theory based on the algebra of relations. This particular approach provides basic and powerful conceptual tools applicable to the problem of automata decomposition. In the second section Muller develops a generalization of automata theory which is an alternative to the generalized automata theory developed by Eilenberg and Wright based on category theory and universal algebra. Muller's generalization is based on formal mathematical notions used in switching and automata theory and provide a convenient way of visualizing the constructions about the systems which enable one to anticipate results.

In last year's report, one section reported research by Putzolu on the subject of asymptotic decomposition of automata. His analysis treated the case in which state based type of realization was assumed and decomposition reporting state splitting was not permitted. In Section 3 recent research results of Putzolu and Muller are presented which consider the case where state splitting is allowed.

Section 4 presents research by C. R. Shepard concerning generation and recognition in formal languages. The research concerns a generalization of the languages generated by a left linear grammar known as equational sets. Equationality and recognizability can be defined for any algebra. The conditions on an algebra such that the definitions of equationality and recognizability coincide are given. If

equational sets of an algebra are recognizable, then all word problems on the algebra are solvable. The converse is false. These abstract results are believed to have some implication on the design of parenthesis free languages, and list structures.

In Section 5 the results of research on the algebraic isomorphism invariants for transition graphs by J. F. Meyer is presented in complete but condensed form. The general results of this research concern 1) How the values of the isomorphism invariants relate to invariant structural characteristics and 2) The specification of special classes of transition graphs for which the isomorphism invariants are complete. The abstract results presented in this section pertain to the problems of code design and automata design.

In Section 6, J. R. Jump's research on the iterative network realization of sequential machines is summarized. A design algorithm is shown to exist for all sequential machines. A specific design algorithm is specified and bound on the complexity of the iterative realization is given for different classes of sequential machines. This study is relevant to the application of lsi technology.

In Section 7, research by E. L. Lawler and M. Moore concerning the sequence of tasks subject to deadline constraints is presented. The class of sequences considered are those for which the consistency principle hold. For this class there exists a linear ordering of tasks which determines an optimal schedule. This principle is exploited to produce solution methods more efficiently than previous methods.

In Section 8, L. Liu presents the structure of formal languages and relates this structure to some of the problems involved in computer languages. Computer languages do not exactly fit the existing classification structure and directions for future research are indicated.

Section 1

A New Approach to Automata Theory

This section presents a new approach to automata theory based on an algebra of relations which is applicable to automata theory in much the same way Boolean algebra is applicable to switching theory.

One of the purposes of this approach is didactic. Previous experience has shown that this method of teaching automata theory is useful with students who have had little contact with the conventional methods and that it provides them with a more powerful way of thinking about the subject.

1.1 An Algebra of Relations

1.1.1 Let Q and Q' be two nonempty sets, and let $Q \times Q'$ represent the Cartesian product or set consisting of all ordered pairs (q, q') , where q is an element of Q and q' is an element of Q' . We may use the formal mathematical notation:

$$Q \times Q' = \{(q, q') \mid q \in Q, q' \in Q'\}.$$

A subset R of such a Cartesian product $Q \times Q'$ is a binary relation. When defining such a binary relation we must give all three sets Q , Q' and R . We shall call Q and Q' the input and output sets respectively and R the set of pairs. Thus, we regard two relations as equal only if they have the same input and output sets as well as the same set of pairs.

There are three other convenient ways that we may think about binary relations. First, if we are given a relation $R \subseteq Q \times Q'$, then we may imagine a Boolean matrix whose rows are labelled by the elements q of Q and whose columns are labelled by the elements q' of Q' . The matrix element in the row q and column q' is made 1 if and only if (q, q') is in R . Otherwise it is made 0. It is clear that for every binary relation we may imagine such a matrix and that conversely for every Boolean matrix we have a corresponding binary relation.

Second, we may think of the relation $R \subseteq Q \times Q'$ as representing a process. If we are given some member q of the input set Q then the process yields an element q' of the output set Q' such that (q, q') is in R . If no such q' exists, then the process must be assumed to produce no output. On the other hand, if several such elements q' exist, then we suppose that the process makes an arbitrary and unpredictable choice of one of them. Since a process must be described more carefully and exactly than has been done here if it is to be useful mathematically, we shall simply regard this model as a convenient intuitive way of thinking about relations.

Third, we may think of a relation as being described by a directed linear graph. This model is particularly useful when the relation is square, that is, when $Q = Q'$. The nodes of the graph are then just the elements of Q and for each pair (q, q') in R , we draw a line from q to q' with an arrow pointing toward q' .

It is also possible to represent relations which are not square by means of linear graphs. Suppose that $R \subseteq Q \times Q'$ is not square, i. e., $Q \neq Q'$. Then we may form a corresponding square relation $R \subseteq (Q \cup Q') \times (Q \cup Q')$. In this case we use the union $Q \cup Q'$ as both the input and output sets. The directed linear graph corresponding to this square relation may also be used to describe the original relation provided the input and output sets are specified.

Binary relations are sets and therefore we may use Boolean algebra when dealing with them. Thus, if $R \subseteq Q \times Q'$ and $S \subseteq Q \times Q'$ are two relations with the same input and output sets, then we may form their set theoretical union $R \cup S \subseteq Q \times Q'$ and intersection $R \cap S \subseteq Q \times Q'$. The union and intersection are both taken as relations having the same input and output sets as R and S . Furthermore, any relation $R \subseteq Q \times Q'$ may be complemented giving $\bar{R} = Q \times Q' - R$ which consists of all pairs (q, q') in $Q \times Q'$ but not in R . We also note that relations with given input and output sets are partially ordered by set inclusion, written $R \subseteq S$.

Relations are quantities with more mathematical structure than sets, and therefore we may define other important operations on relations. If $R \subseteq Q \times Q'$ and $S \subseteq Q' \times Q''$ are two relations such that the output set Q' of R is the same as the input set of S , then we may define the Pierce product $R \circ S$ of R by S . We take $R \circ S \subseteq Q \times Q''$ as the relation consisting of all pairs (q, q'') such that for some q' in Q' we have (q, q') in R and (q', q'') in S . We may use the formal

mathematical notation:

$$R \circ S = \{(q, q'') \mid \exists q' \in Q', (q, q') \in R, \text{ and } (q', q'') \in S\}.$$

If R and S are regarded as corresponding to Boolean matrices, then $R \circ S$ corresponds to a form of Boolean matrix product in which the summation which is used in the conventional definition of matrix product is replaced by the Boolean operation "or". Similarly, if R and S are thought of as representing processes, then $R \circ S$ represents a combined process in which the output of R is used as the input for S .

We further define the inverse or transpose R^{-1} of the relation $R \subseteq Q \times Q'$ as consisting of all pairs (q', q) such that (q, q') is in R . Thus, R^{-1} is a subset of $Q' \times Q$ so that the input and output sets are interchanged when taking the transpose. In the case of the linear graph model, the transpose is formed by reversing the directions of the arrows of the graph.

We will now show a precise correspondence between identities on Boolean matrices and identities on matrices of integers.

Let Z be the set of nonnegative integers and define a mapping β from Z to the set $\{0, 1\}$ by the rule that β maps 0 to 0 and β maps i to 1 if $i > 0$. If M and N are matrices whose elements are nonnegative integers, then let M' and N' be the corresponding Boolean matrices obtained by replacing the elements of M and N respectively with their images under the mapping β . We may easily derive the following rules which allow us to regard β as a kind of homomorphism.

1. $(M+N)' = M' \cup N'$
2. $(M-N)' = M' \circ N'$
3. $(M \odot N)' = M' \cap N'$
4. $(M^T)' = (M')^{-1}$
5. If $M \leq N$, then $M' \subseteq N'$.

Here, the product $M \odot N$ means elementwise multiplication of the two matrices and M^T represents the transpose. Yet a sixth rule might be written for multiplication of a matrix by a scalar.

The preservation of the various operations under the homomorphism β allows us to transform any identity involving matrices of non-negative integers and these operations into an identity for Boolean matrices.

Not all operations on matrices of nonnegative integers are preserved under β . For example, the matrix difference $M-N$ does not map uniformly, and identities involving the difference do not generally correspond to identities on Boolean matrices. Furthermore, since the mapping is not one-to-one there may be identities which are valid for Boolean matrices but which do not hold for matrices of nonnegative integers.

1. 1. 2 In our study of the properties of relations we shall need to consider relations with certain special properties, and we therefore proceed to define terms for some of these properties. Let $R \subseteq Q \times Q'$ be a relation, then:

1. R is determinate iff for each $q \in Q$ there is at most one $q' \in Q'$ such that (q, q') is in R .
2. R is productive iff for each $q \in Q$ there is at least one $q' \in Q'$ such that (q, q') is in R .
3. R is a single valued iff it is both determinate and productive. It is then called a mapping or a function from Q to Q' .
4. R is surjective or onto iff for each $q' \in Q'$ there is at least one $q \in Q$ such that (q, q') is in R .
5. R is injective iff it is single valued and R^{-1} is determinate.
6. R is one-to-one iff R and R^{-1} are both single valued.

Using the matrix model, we see that if R is determinate, then each row of the corresponding matrix may have at most one element which is 1. Similarly, if R is productive then each row contains at least one 1.

In our model of R as a process we see that if R is determinate then the result q' if it exists, is uniquely determined by the input q . Similarly, if R is productive then some output q' must occur for every input.

We shall write ϕ to denote an empty relation. The set ϕ is regarded as a subset of $Q \times Q'$ but containing no pairs. This special relation has the properties of a zero with respect to Pierce product. That is, $R \circ \phi = \phi \circ R = \phi$ whenever such multiplication is possible with a relation R . The Boolean matrix corresponding to ϕ is a matrix whose elements are all 0.

If Q is a set, then define $\langle Q$ as the set of all pairs $(1, q)$, where q is in Q and 1 is the element of a singleton set 1 . Thus, $\langle Q$ may be regarded as a row vector whose elements are all 1 . The transpose of $\langle Q$ we shall write as $Q \rangle$ and it consists of all pairs $(q, 1)$ where q is in Q . We note that $Q \rangle$ is a column vector. A similar notation may be used for subsets of Q . Thus, if $K \subseteq Q$ then $\langle K$ and $K \rangle$ represent row and column vectors respectively whose elements are 1 precisely for members of K .

Row and column vectors may be used with the Pierce product notation. In particular, if $R \subseteq Q \times Q'$ is any relation, then $\langle Q \circ R$ is called the range of R and $R \circ Q' \rangle$ is called the domain of R . We note also that $Q \rangle \circ \langle Q' = Q \times Q'$, providing a convenient abbreviation.

1. 1. 3 We now direct our attention to the case of square relations. A particular square relation $I_Q \subseteq Q \times Q$ of some importance is called the identity on Q . It consists of just those pairs (q, q') for which $q = q'$. Thus, $I_Q = \{(q, q) \mid q \in Q\}$. The Boolean matrix corresponding to I_Q consists of 1 's along the diagonal and 0 's everywhere else.

Let $R \subseteq Q \times Q$ be a square relation, then:

7. R is symmetric iff whenever (q, q') is in R , then (q', q) is also in R .
8. R is antisymmetric iff whenever (q, q') is in R and $q \neq q'$, then (q', q) is not in R .
9. R is reflexive iff for each $q \in Q$, the pair (q, q) is in R .

10. R is irreflexive iff for each $q \in Q$, the pair (q, q) is not in R .
11. R is weakly reflexive iff whenever (q, q') is in R , then (q', q') is in R .
12. R is dually weakly reflexive iff whenever (q, q') is in R , then (q, q) is in R .
13. R is transitive iff whenever (q, q') and (q', q'') are both in R , then (q, q'') is in R .
14. R is an equivalence relation iff it is reflexive, symmetric, and transitive.
15. R is a partial order iff it is reflexive, antisymmetric, and transitive.
16. R is a total order iff it is a partial order and every pair (q, q') in $Q \times Q$ is either in R or in R^{-1} .

Using our matrix model, we see that if R is weakly reflexive then any column in the corresponding matrix which is not all 0 has a diagonal element which is 1. In the graph corresponding to a weakly reflexive relation, any node having an arrow pointing toward it must also have a self loop.

The terms "antisymmetric" and "irreflexive" are subject to misinterpretation. There are relations, such as I_Q , which are both symmetric and antisymmetric, so that antisymmetry is not the denial of symmetry. Also, some relations are neither symmetric nor antisymmetric and some relations are neither reflexive nor irreflexive.

Many of the special properties we have defined have formal interpretations. Examples of such interpretations are contained in the following list in which it is assumed that R has the property in question.

1. Determinate: $R^{-1} \circ R \subseteq I_Q$.
2. Productive: $R \circ R^{-1} \supseteq I_Q$, or alternatively $R \circ Q' \supseteq Q$.
3. Symmetric: $R = R^{-1}$.
4. Antisymmetric: $R \cap R^{-1} \subseteq I_Q$.
5. Reflexive: $R \supseteq I_Q$.
6. Irreflexive: $\bar{R} \supseteq I_Q$, or alternatively $R \cap I_Q = \phi$.
7. Weakly reflexive: $R \circ (R \cap I_Q) = R$.
8. Dually weakly reflexive: $(R \cap I_Q) \circ R = R$.
9. Transitive: $R \circ R \subseteq R$.

1.1.4 If $R \subseteq Q \times Q$ is any square relation, then although it may not be reflexive, we may obtain a reflexive relation from it by the simple device of forming $R \cup I_Q$. This new relation $R \cup I_Q$ will be called the reflexive extension of R and may be constructed using our matrix model by simply replacing each diagonal element of R by 1.

Similarly, R may not be symmetric, but we may use R to form the relation $R \cup R^{-1}$ which is symmetric. It is also true that $R \cap R^{-1}$ is symmetric, but we shall define the symmetric extension of R to be $R \cup R^{-1}$ rather than $R \cap R^{-1}$.

Finally, we define $R \cup R^{-1} \cup I_Q$ as the symmetric and reflexive extension of R . We note that it is both the symmetric extension of the reflexive extension and the reflexive extension of the symmetric extension. Clearly, $R \cup R^{-1} \cup I_Q$ is both symmetric and reflexive.

Somewhat less obvious is how one might construct a transitive extension of a square relation R . This extension, however, will be very important in our later analysis and for this reason we designate it by the special symbol R^+ . We define R^+ as the relation consisting of all pairs (q, q') such that for some finite sequence q_1, q_2, \dots, q_n of elements of Q we have $q = q_1$ and $q_n = q'$ while each consecutive pair (q_i, q_{i+1}) of terms in the sequence is a member of R . We do not require that the terms of the sequence all be distinct, but we do assume that the number n of terms is at least 2.

A formal representation of the above definition is:

$$R^+ = R \cup R \circ R \cup R \circ R \circ R \dots$$

In the definition of the transitive extension if we had allowed n to be 1 as well as larger integers, then we obtain a relation which will be designated by R^* . It is called the reflexive and transitive extension of R . One may show quite easily that $R^* = R^+ \cup I_Q$ and that $R^+ = R \circ R^*$. Thus, R^+ and R^* are transitive but R^+ may not be reflexive while R^* is always reflexive.

We define $R \subseteq Q \times Q$ to be acyclic if R^+ is irreflexive. Using the model of a directed linear graph we have a convenient interpretation for this property. We note from our definition of R^+ that it consists

of all pairs (q, q') which are joined by a chain of sequence q_1, q_2, \dots, q_n which starts at $q = q_1$ and ends at $q_n = q'$. Thus, if R is irreflexive we can never have $q = q'$. In other words, there can be no loops or cycles in the graph which permit one to pass from some point back to itself following the direction of the arrows.

A familiar example of an acyclic relation is provided by the concept of a combinational circuit. In such a circuit we let Q represent the set of nodes of the circuit and place (q, q') in R whenever there is a switching element having q for an input and q' for an output. Thus, if the circuit is combinational there is no feedback and hence no cycles of the type described. This is equivalent to saying that R is acyclic.

1. 1. 5 If $R \subseteq Q \times Q$ is any square relation, then we define the trace of R which is written $\text{Tr}(R)$ to be 1 if (q, q) is in R for some $q \in Q$ and to be 0 otherwise. In terms of our matrix model we form $\text{Tr}(R)$ by taking the union of the diagonal elements of the matrix corresponding to R . Also, in the linear graph model we note that $\text{Tr}(R) = 1$ if the graph corresponding to R has any self loops and $\text{Tr}(R) = 0$ otherwise.

If we regard 1 as standing for truth and 0 as standing for falsehood, then $\text{Tr}(R)$ represents the proposition $(\exists q \in Q, (q, q) \in R)$.

In the case of matrices whose elements are integers, it is customary to define the trace of a matrix as the sum of its diagonal elements. Since our mapping $\beta: \mathbb{Z} \rightarrow \{0, 1\}$ is a homomorphism if summation becomes union, we see that the trace operation is preserved

under this mapping. That is to say, the image of $\text{Tr}(M)$ under β is just $\text{Tr}(M')$ if M' is the image of M .

1. 1. 6 It is easily shown that if R is any relation, then $R \circ R^{-1}$ is square, symmetric, and weakly reflexive. We therefore may wonder if whenever we have a relation $S \subseteq Q \times Q$ which is square, symmetric, and weakly reflexive, whether it can be written in the form $R \circ R^{-1} = S$ for some relation R . This question will be answered in the affirmative, but some proof is required.

Let $S \subseteq Q \times Q$ be given as above and define a family \mathcal{K} of subsets K of Q by placing K in \mathcal{K} iff it satisfies the following property for every $q \in Q$. The element q is in K iff (q, q') is in S for each $q' \in K$. A formal definition of \mathcal{K} may be written as follows:

$$\mathcal{K} = \{K \subseteq Q \mid \forall q \in Q, [q \in K \iff \forall q' \in K, (q, q') \in S]\}.$$

Although this definition of the family \mathcal{K} is complicated we gain some insight into its nature and structure by developing some of its properties.

Define a subset J of Q to be complete iff every pair (q, q') of not necessarily distinct elements of J is also a pair in S . A maximal complete set is one which is not properly contained in any other complete set.

Lemma 1

\mathcal{K} is the family of all maximal complete sets.

Proof: If K is in \mathcal{K} and q is in K , then $(q, q') \in S$ for each $q' \in K$.

Hence every pair (q, q') of not necessarily distinct elements of K

must be a pair in S , so K is complete. In fact, we see that this property expresses the "only if" part of our defining condition for \mathcal{K} .

Next, consider a complete set J which is not maximal. There must be some other complete set J' such that $J \subsetneq J'$. Therefore, there is at least one element $q \in J'$ such that q is not in J . Since J' is complete, we see that (q, q') is in S whenever q' is in J . Thus, J cannot be in \mathcal{K} and every set $K \in \mathcal{K}$ must be a maximal complete set. We see that this latter property expresses the "if" part of our defining condition for \mathcal{K} . Hence, the defining condition is satisfied by every maximal complete set, thus proving the lemma.

Lemma 2

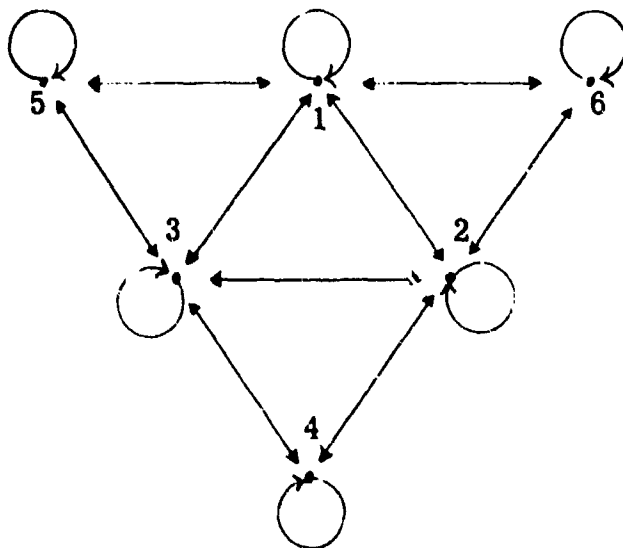
Whenever (q, q') is a pair in S , there is some $K \in \mathcal{K}$ such that both q and q' are in K .

Proof: If $q = q'$, then the singleton set $\{q\}$ is complete. Otherwise, if $q \neq q'$, then the two element set $\{q, q'\}$ is complete because (q', q) , (q, q) and (q', q') must all be in S as a result of the fact that S is symmetric and weakly reflexive. Thus, q and q' are both in some complete set C_1 . Let \mathcal{C} be the family of all complete sets C which include C_1 . Then \mathcal{C} is partially ordered under set inclusion and fulfills the condition of Zorn's lemma, namely that every chain in \mathcal{C} have an upper bound. Hence, we conclude from Zorn's lemma that \mathcal{C} has a maximal element K . This element is also maximal in the family of all complete sets since any set including K must also include C_1 . Hence K is in \mathcal{K} by lemma 1, and we have completed our proof.

Remark 1

There may be other families of maximal complete sets which have the property expressed in lemma 2.

Proof: We give a simple example of such a case. Let Q be a six element set $\{1, 2, 3, 4, 5, 6\}$ and let S contain the pairs $(1, 2)$, $(2, 3)$, $(3, 1)$, $(2, 4)$, $(3, 4)$, $(1, 5)$, $(3, 5)$, $(1, 6)$, $(2, 6)$ as well as all other pairs required to satisfy symmetry and reflexivity. Then, the graph has the following form:



The family of three maximal complete sets $\{1, 2, 6\}$, $\{2, 3, 4\}$, $\{1, 3, 5\}$ satisfies the condition of lemma 2 but it does not contain the maximal complete set $\{1, 2, 3\}$. Thus, the remark is proved.

Continuing with our original development, we introduce a set W , called the normal set which is in one-to-one correspondence with the family \mathcal{K} . We write $w \sim K$ to indicate that the element $w \in W$

corresponds to the set $K \in \mathcal{K}$. This set W is to be used as a notational convenience in describing properties of the relation S . When it is necessary to specify S we shall use subscript notation as W_S and \mathcal{K}_S .

Define the relation $S^d \subseteq Q \times W$ by the rule that (q, w) is in S^d iff $q \in K$, where $w \sim K$. We shall write S^{-d} as an abbreviation for $(S^d)^{-1}$.

Theorem 1

If $S \subseteq Q \times Q$ is symmetric and weakly reflexive,
then $S = S^d \circ S^{-d}$.

Proof: If (q, q') is in S then by lemma 2 there is a set K of \mathcal{K} containing both q and q' . Pick $w \in W$ which corresponds to this set. Since (q, w) is in S^d and (w, q') is in S^{-d} , we see that (q, q') is in $S^d \circ S^{-d}$. Hence, $S \subseteq S^d \circ S^{-d}$.

Next, suppose that (q, q') is in $S^d \circ S^{-d}$. Then, there must be some element $w \in W$ such that (q, w) is in S^d and (w, q') is in S^{-d} . Pick $K \in \mathcal{K}$ corresponding to this w . Since both q and q' are in K we have $(q, q') \in S$ by lemma 1, and the theorem is proved.

Remark 2

If S is transitive then S^d is determinate and the family \mathcal{K} is pairwise disjoint.

Proof: Let K and K' be two not necessarily distinct elements of \mathcal{K} and assume $K \cap K' \neq \emptyset$. If q_1 is in $K \cap K'$ then (q, q_1) is in S for all $q \in K$ since K is complete and similarly (q_1, q') is in S for all $q' \in K'$.

Since S is transitive, we see that (q, q') must always be in S whenever q is in K and q' is in K' . But, S is symmetric and weakly reflexive, and therefore $K \cup K'$ is complete. However, K and K' are maximal complete sets so therefore $K = K'$. Therefore, if $K \neq K'$, we have $K \cap K' = \emptyset$ and \mathcal{K} is pairwise disjoint.

To show that S^d is determinate we consider any element $q \in Q$. If (q, w) is in S^d , then q is in the set $K \in \mathcal{K}$ which corresponds to $w \in W$. But, q can be in no other set K' of \mathcal{K} so there is no other element w' of W such that (q, w') is in S^d . Hence S^d is determinate and the remark is proved.

Remark 3

If S is reflexive then S^d is productive and \mathcal{K} is a cover of Q in the sense that $\bigcup_{K \in \mathcal{K}} K = Q$.

Proof: Let q be any element of Q . Then (q, q) is in S since S is reflexive so the singleton set $\{q\}$ is complete. By lemma 2, there is some $K \in \mathcal{K}$ such that $q \in K$, and hence \mathcal{K} is a cover. Also if $w \sim K$, then (q, w) is in S^d so S^d is productive.

Remark 4

If S is an equivalence relation then S^d is single valued and \mathcal{K} is a partition of Q .

Proof: The conditions of both remarks 2 and 3 are satisfied if S is an equivalence relation since it is then both reflexive and transitive. Hence \mathcal{K} is a partition, i.e., a pairwise disjoint cover. Also S^d is determinate and productive, thus single valued.

We note that the sets of are called equivalence sets when S is an equivalence relation.

1. 1. 7 Define the relation $R \subseteq Q \times Q'$ to be finitary iff there is a finite set Q'' which admits two relations $G \subseteq Q \times Q''$ and $H \subseteq Q'' \times Q'$ such that $R = G \circ H$. We note that any finite relation R must be finitary since $R = R \circ I_{Q'}$.

Theorem 2

If $S \subseteq Q \times Q$ is symmetric and weakly reflexive,
then S is finitary iff the corresponding family
is finite.

Proof: If \mathcal{K} is finite then W is finite and $S = S^d \circ S^{-d}$, where $S^d \subseteq Q \times W$ and $S^{-d} \subseteq W \times Q$ so S is finitary.

Next, suppose that S is finitary and hence may be written in the form $S = G \circ H$ where $G \subseteq Q \times Q''$ and $H \subseteq Q'' \times Q$ for finite Q'' . Define an equivalence relation $E \subseteq Q \times Q$ by the rule that (q_1, q_2) is in E iff for all $q'' \in Q''$ we have $(q_1, q'') \in G \iff (q_2, q'') \in G$. It is trivial to show that E is an equivalence relation. Also, the number of equivalence sets is finite since if Q'' is of cardinality n the number of equivalence sets can be no more than 2^n . We further see that if (q_1, q_2) is in E , then for all $q \in Q$ we have $(q_1, q) \in S \iff (q_2, q) \in S$. Thus, q_1 and q_2 will always be in exactly the same sets K of since the rule for inclusion or exclusion applies equally to both and S is weakly reflexive so that inclusion of one will not cause exclusion of the other. This means that each K of is a union of E equivalence sets.

Thus, \mathcal{R} is finite since the cardinality of \mathcal{R} can be no greater than the number of possible unions of E equivalence sets which may be formed.

Define the weight of a finitary relation $R \subseteq Q \times Q'$ to be the minimum cardinality of any set Q'' such that $R = G \circ H$ for some $G \subseteq Q \times Q''$ and $H \subseteq Q'' \times Q'$. Further, let R^2 be an abbreviation for $R \circ R$, and in general, R^n for the n -fold Pierce product of R .

Theorem 3

If $R \subseteq Q \times Q$ is finitary and of weight n , then

$$R^+ = R \cup R^2 \cup \dots \cup R^n.$$

Proof: From the definition of R^+ , we know that $R^+ \supseteq R \cup R^2 \cup \dots \cup R^n$. Hence, to show equality it is only necessary to prove that any pair $(q, q') \in R^+$ is also in $R \cup R^2 \cup \dots \cup R^n$. If (q, q') is in R^+ , then (q, q') is in R^m for some integer m , and we may take $m > n$ for otherwise the conclusion is obvious. Write $R = G \circ H$ as above, where $G \subseteq Q \times Q''$ and $H \subseteq Q'' \times Q$, and Q'' has cardinality n . Then, $R^m = G \circ H \circ \dots \circ G \circ H = G \circ (H \circ G)^{m-1} \circ H$, where $H \circ G \subseteq Q'' \times Q''$ and we adopt the convention that $(H \circ G)^0 = I_{Q''}$. Since (q, q') is in R^m , there must be a pair $(p, p') \in (H \circ G)^{m-1}$ such that (q, p) is in G and (p', q') is in H . However, if (p, p') is in $(H \circ G)^{m-1}$, there is a sequence $p = p_1, p_2, \dots, p_m = p'$ such that each consecutive pair (p_i, p_{i+1}) is in $H \circ G$. But, Q'' contains only n elements and $m > n$ so the terms in the sequence cannot all be distinct. We can therefore contract the sequence, for if $p_i = p_j$, then $p = p_1, p_2, \dots, p_i, p_{j+1},$

..., $p_m = p'$ satisfies the required condition on consecutive pairs. Hence there is some integer $m' \leq n$ such that (p, p') is in $(H \circ G)^{m'-1}$ and therefore (q, q') is in $G \circ (H \circ G)^{m'-1} \circ H = R^{m'}$. This completes the proof of the theorem.

Theorem 4

If $R \subseteq Q \times Q'$ is finitary, then $\bar{R} = Q \times Q' - R$ is finitary.

Proof: Let us write $R = G \circ H$, where $G \subseteq Q \times Q''$ and $H \subseteq Q'' \times Q'$, with Q'' finite. Let $\mathcal{P}(Q'')$ be the family of all subsets of Q'' . Construct a relation $G' \subseteq Q \times \mathcal{P}(Q'')$ by placing (q, P) in G' iff P consists of just those elements $q'' \in Q''$ such that (q, q'') is in G . Evidently, G' is single valued. Also, construct a relation $H' \subseteq \mathcal{P}(Q'') \times Q'$ by placing (P, q') in H' iff (q'', q') is not in H whenever q'' is in P . Consider the product $G' \circ H'$. If (q, q') is in $G' \circ H'$, then for the unique P associated with q , we have (P, q') not in H so (q, q') is not in $G \circ H = R$. On the other hand, if (q, q') is not in $G' \circ H'$, then for the unique P associated with q , we have (P, q') not in H' . Thus, there is some $q'' \in P$ such that (q'', q') is in H . Hence, (q, q') is in $G \circ H = R$. Therefore, $G' \circ H' = \bar{R}$ and since Q'' is finite, $\mathcal{P}(Q'')$ is also finite, showing that \bar{R} is finitary.

1.2 Relational Systems

1.2.1 To describe a relational system S , one must specify:

- (1) An alphabet A which is a finite set of letters a which are symbols used for indexing and labelling expressions.
- (2) A carrier set Q consisting of states or elements q of the system.
- (3) A mapping F from A to the family of square relations on Q .

The image of $a \in A$ under F will be written F_a and called the set of transitions under a . Thus, $F_a \subseteq Q \times Q$.

We may think of a relational system S as being equal to the triple (A, Q, F) . It may be represented by a labelled graph whose nodes correspond to the states q of Q . Between every pair (q, q') of states in F_a we draw an arrow pointing toward q' which we label with the letter a . Such a graph is sometimes called the state diagram of a nondeterministic automaton.

Two relational systems $S = (A, Q, F)$ and $S' = (A, Q', F')$ which use the same alphabet A are called similar systems. We shall be principally concerned with the interaction of similar systems and this is our reason for setting up such an elaborate structure. If one were concerned just with the properties of a single system $S = (A, Q, F)$ he could use the letters a in A to stand for the corresponding relations F_a . Thus, the mapping F would not need to be explicitly mentioned because it would be placed on a higher level of abstraction than is now being contemplated. However, this cannot be our present approach.

Important special cases are now listed.

1. We call S determinate iff every relation F_a is determinate.
2. We call S productive iff every relation F_a is productive.
3. We call S single valued iff every relation F_a is single valued.

The case in which S is single valued has been extensively investigated in the literature. The following names have been given to this type of relational system: Monadic algebra (J. B. Wright), Unary algebra (J. R. Buchi), Semi automaton (A. Ginsburg). An important fact which was recognized by Wright and Buchi was that a single valued relational system is an algebra in the sense of Garrett Birkhoff. Therefore, a host of theorems which apply generally to algebras also hold for these relational systems.

1.2.2 Again, let us consider a general relational system $S = (A, Q, F)$. We write \mathcal{F} to represent the family of all relations F_a , where a is in A . Also, let \mathcal{F}^+ stand for the closure of \mathcal{F} under Pierce product. The input semigroup of S is the system whose carrier is \mathcal{F}^+ and whose operation is Pierce product. We see that this system is indeed a semigroup since Pierce product is associative and \mathcal{F}^+ is closed under it.

It was apparent from section 1.1.3 that I_Q has the properties of a unit. Thus for all relations $F_t \in \mathcal{F}^+$ we have $I_Q \circ F_t = F_t \circ I_Q = F_t$. If we attach I_Q to \mathcal{F}^+ we get a set $\mathcal{F}^* = \mathcal{F}^+ \cup \{I_Q\}$ which is the carrier of the input monoid of S . A monoid is defined as a semigroup with a unit and we see that \mathcal{F}^* forms a monoid under Pierce product.

Theorem 5

To every monoid M with a finite set of generators G ,
there corresponds a single valued relational system

$$S_M = (G, M, F) \text{ whose input monoid is isomorphic to } M.$$

Proof: From the statement of the theorem, we see that G is to be taken as the alphabet of S and M as its carrier. For each element $t \in M$, we define F_t as the set of all pairs (s, s') such that $st = s'$. The restriction of t to G provides us with the definition for F of S_M . Clearly, S_M is single valued. We now show that the correspondence $t \sim F_t$ is one-to-one. Let e stand for the unit of M , then (e, t) is in F_t but not in $F_{t'}$, if $t \neq t'$. Thus, if $F_t = F_{t'}$, we have $t = t'$. Also, $F_s \circ F_t = F_{st}$ since M is associative. This shows that M is isomorphic to the family $\{F_t\}$ under the operation of Pierce product. Finally, we prove that \mathcal{F}^* is the same as the family $\{F_t\}$. However, F_g is in \mathcal{F}^* for each $g \in G$ and G is a set of generators for M . Thus, each element $t \in M$ is either the unit e or representable in one or more ways as finite product $g_1 g_2 \dots g_n$ of generators. In the former case, we have $F_e = I_M \in \mathcal{F}^*$ and in the latter case $F_{g_1 g_2 \dots g_n} = F_{g_1} \circ F_{g_2} \circ \dots \circ F_{g_n} \in \mathcal{F}^*$. Hence, \mathcal{F}^* is just the family $\{F_t\}$, where $t \in M$, and the theorem is proved.

1.2.3 A finite sequence $a(1) a(2) \dots a(n)$ of not necessarily distinct letters taken from an alphabet A is also sometimes called a string or word or tape on the alphabet A . Let A^+ represent the set of all such strings on A . Two strings in A^+ will be considered to be equal

if the letters occurring in corresponding positions of the strings are always the same. We define the length of a string $a(1) a(2) \dots a(n)$ to be the number n of letter which make it up.

The concatenation of two strings $s = a(1) \dots a(m)$ and $t = b(1) \dots b(n)$ is the new string $st = a(1) \dots a(m) b(1) \dots b(n)$ formed by juxtaposing the two. Clearly, concatenation is an associative operation. Also, A^+ is closed under concatenation and we call this system the free semigroup generated by A.

One may attach a unit element e to A^+ , giving a set $A^* = A^+ \cup \{e\}$. We define e to be an element which follows the operation rules $et = te = t$ for all t in A^* . Although it is difficult to imagine e to be a string or sequence, we shall call it the string of length zero. It has also been variously called the null sequence, the empty word, and possibly other names. The set A^* under concatenation and the unit rules for e , we call the free monoid generated by A.

Using theorem 5, we may construct $A_* = (A, A^*, F)$ which is called the singly generated, free monadic algebra on A. The carrier A^* of this relational system is the same as the carrier of the free monoid from which it is derived. Each relation F_a consists of all pairs (t, ta) , such that t is a string in A^* .

Other definitions concerning A^* will be needed in our later work. We call a string $r \in A^*$ a prefix of a string $s \in A^*$ iff s can be written in the form $s = rt$ for some string $t \in A^*$. We note that if the string t exists it is unique. Hence we define r/s to be t if $s = rt$ and take it

to be undefined if no such t exists. Similarly, we say t is a suffix of s iff $s = rt$ for some r . We also introduce the notation $s/t = r$ as in the previous definition.

A proper prefix of a string s is a prefix of s which is shorter than s . Similarly define a proper suffix. A segment of s is a prefix of a suffix of s . It is a proper segment if it is shorter than s . We note that a segment of s is also a suffix of a prefix of s . In fact, s' is a segment of s when we have $s = rs't$ for some r and t . We might then use the notation $s' = r \backslash s / t$.

The approach which we are taking to the algebra of strings in this section may be described as an informal one. We simply set down some of the properties of strings and rely on the experience of the reader to justify their reasonableness.

Others have taken more formal approaches which require more intellectual effort and are not necessary for our purposes. One such approach is to base the algebra of strings on a simple set of postulates which are similar to the Peano postulates for numbers. We can then derive such rules as the associative law for concatenation.

Another approach is to treat each string as a sequence in the sense of being a mapping from a set of consecutive integers to A . Then the mechanism of concatenation is described in terms of such mappings and the results are proved.

1. 2. 4 Let $S = (A, Q, F)$ and $S' = (A, Q', F')$ be two similar relational systems which may or may not be distinct. Then a relation

$\sigma \subseteq Q \times Q'$ is called a generalized congruence between S and S' iff for all $a \in A$,

- i) $F_a^{-1} \circ \sigma \circ F_a' \subseteq \sigma$,
- ii) $\sigma \circ F_a' \circ Q' \supseteq F_a \circ Q$, and
- iii) $\sigma^{-1} \circ F_a \circ Q \supseteq F_a' \circ Q'$

This formal definition can also be expressed verbally. Property (i) is called the substitution property and corresponds to the statement that if (q_1, q_2') is a pair in σ and if (q_1, q_2) is in F_a and (q_1', q_2') is in F_a' then (q_2, q_2') is also in σ . Properties (ii) and (iii) are dual in the sense that one may be obtained from the other by interchanging S and S' while taking the inverse of σ . Thus, if σ is symmetric, the last two properties are identical. We may express (ii) by saying that whenever (q_1, q_1') is in σ and there is a pair $(q_1', q_2') \in F_a'$, then there is also a pair $(q_1, q_2) \in F_a$. This property is automatically fulfilled if S is productive. Similarly (iii) holds automatically if S' is productive. Thus, if only productive systems are being considered we need merely state (i).

A number of special cases of a generalized congruence σ are now listed.

1. σ is a congruence if it is an equivalence relation.
2. σ is a homomorphism if it is single valued. Special types of homomorphism are now listed.
3. σ is a monomorphism if σ^{-1} is determinate.
4. σ is an epimorphism if σ^{-1} is productive.

5. c is an endomorphism if it is square.
6. σ is an isomorphism if σ^{-1} is single valued.
7. σ is an automorphism if it is a square isomorphism.

Theorem 6

If $\sigma_1 \subseteq Q \times Q'$ is a generalized congruence between S and S' and $\sigma_2 \subseteq Q' \times Q''$ is a generalized congruence between S' and S'' , then $\sigma_1 \circ \sigma_2$ is a generalized congruence between S and S'' .

Proof: To prove $F_a^{-1} \circ \sigma_1 \circ \sigma_2 \circ F_a'' \subseteq \sigma_1 \circ \sigma_2$, we assume (q_2, q_2'') is an element of $F_a^{-1} \circ \sigma_1 \circ \sigma_2 \circ F_a''$. Then find q_1, q_1' , and q_1'' such that: $(q, q_2) \in F_a$, $(q_1, q_1') \in \sigma_1$, $(q_1', q_1'') \in \sigma_2$, and $(q_1'', q_2'') \in F_a''$. Then, by $\sigma_2 \circ F_a'' \circ Q'' > \subseteq F_a' \circ Q' >$ we see that since $(q_1', 1)$ is in $\sigma_2 \circ F_a'' \circ Q'' >$, it is in $F_a' \circ Q' >$. Hence there is a $q_2' \in Q'$ such that (q_1', q_2') is in F_a' . Now, $F_a^{-1} \circ \sigma_1 \circ F_a' \subseteq \sigma_1$, so (q_2, q_2') is in σ_1 . Similarly, (q_2', q_2'') is in σ_2 , thus proving that (q_2, q_2'') is in $\sigma_1 \circ \sigma_2$.

To prove $\sigma_1 \circ \sigma_2 \circ F_a'' \circ Q'' > \subseteq F_a \circ Q >$, we proceed algebraically. Use $\sigma_2 \circ F_a'' \circ Q'' > \subseteq F_a' \circ Q' >$ and obtain $\sigma_1 \circ \sigma_2 \circ F_a'' \circ Q'' > \subseteq \sigma_1 \circ F_a' \circ Q' >$. Since $\sigma_1 \circ F_a' \circ Q' > \subseteq F_a \circ Q >$, the result follows. The third property for $\sigma_1 \circ \sigma_2$ is proved in a similar way.

1.2.5 Let $S = (A, Q, F)$ be a relational system. We may extend our notation by defining F_s for each $s \in A^*$ as follows. When $s = e$, then let $F_e = I_Q$. When s is a string $a(1) a(2) \dots a(n)$ of length $n > 0$,

then let $F_s = F_{a(1)a(2)\dots a(n)} = F_{a(1)} \circ F_{a(2)} \circ \dots \circ F_{a(n)}$. Using this rule, we see that whenever s and t are two strings in A^* , then $F_{st} = F_s \circ F_t$. Consequently, \mathcal{F}^* consists of just those relations which can be represented in the form F_s , where s is in A^* . We have thus extended F so as to map A^* onto \mathcal{F}^* .

Theorem 7

If σ is a generalized congruence between $S = (A, Q, F)$ and $S' = (A, Q', F')$, then for each $s \in A^*$,

- i) $F_s^{-1} \circ \sigma \circ F_s' \subseteq \sigma$,
- ii) $\sigma \circ F_s' \circ Q' > \subseteq F_s \circ Q >$, and
- iii) $\sigma^{-1} \circ F_s \circ Q > \subseteq F_s' \circ Q' >$.

Proof: The proof is by induction on the length of the string s .

Consider first the case in which $s = e$, so $F_e = I_Q$ and $F_e' = I_{Q'}$.

Then (i), (ii), and (iii) are:

- i) $I_Q \circ \sigma \circ I_{Q'} \subseteq \sigma$,
- ii) $\sigma \circ I_{Q'} \circ Q' > \subseteq I_Q \circ Q >$, and
- iii) $\sigma^{-1} \circ I_Q \circ Q > \subseteq I_{Q'} \circ Q' >$.

These are all trivial.

Next, suppose that (i), (ii), and (iii) hold for some string $s \in A^*$ and let a be any letter in A . We then seek to prove these same three properties for the string sa . Now, $F_{sa} = F_s \circ F_a$ and $F_{sa}' = F_s' \circ F_a'$, so

- i) $F_{sa}^{-1} \circ \sigma \circ F_{sa}' = F_a^{-1} \circ F_s^{-1} \circ \sigma \circ F_s' \circ F_a' \subseteq F_a^{-1} \circ \sigma \circ F_a' \subseteq \sigma$.

To prove

ii) we note that $F_a^{-1} \circ Q' > \underline{\quad} Q' >$, so $\sigma \circ F_s' \circ Q' >$

$$\subseteq \sigma \circ F_s' \circ Q' > \subseteq F_s \circ Q > .$$

Thus, if $(q, 1)$ is in $\sigma \circ F_s' \circ F_a' \circ Q' >$, there must be a pair $(q, q_1) \in F_s$. Also there are pairs $(q, q') \in \sigma$ and $(q', q_1') \in F_s'$. Hence (q_1, q_1') is in σ by hypothesis (i) where q_1' is in $F_a' \circ Q' >$. Thus $(q_1, 1)$ is in $\sigma \circ F_a' \circ Q' >$, and hence in $F_a \circ Q >$. This shows that $(q, 1)$ is in $F_s \circ F_a \circ Q >$. We prove (iii) similarly. This completes the inductive step and hence the proof.

Theorem 3

If $\sigma \subseteq Q \times Q$ is a generalized congruence on $S = (A, Q, F)$ which is symmetric and weakly reflexive, then $\sigma^d \subseteq Q \times W$ is a generalized congruence between S and a system $S^W = (A, W, F^W)$ such that for each $a \in A$, the relation F_a^W is defined by $F_a^W = \overline{(\sigma^{-1} \circ F_a \circ \sigma^d)} \cap \sigma^{-d} \circ F_a \circ \sigma^d$.

Proof: We must prove that σ^d has the three properties

- i) $F_a^{-1} \circ \sigma^d \circ F_a^W \subseteq \sigma^d$,
- ii) $\sigma^d \circ F_a^W \circ W > \subseteq F_a \circ Q >$, and
- iii) $\sigma^{-d} \circ F_a \circ Q > \subseteq F_a^W \circ W >$.

The definition of F_a^W may be expressed less formally by saying that (w_1, w_2) is in F_a^W iff whenever (w_1, q) is in $\sigma^{-d} \circ F_a$, then (q, w_2) is in σ^d and furthermore such an element q exists.

To prove (i), let us assume that (q, w_2) is in $F_a^{-1} \circ \sigma^d \circ F_a^W$.

An element w_1 must exist such that (w_1, q) is in $\sigma^{-d} \circ F_a$ and (w_1, w_2)

is in F_a^W . Hence, we see that (q, w_2) is in σ^d and (i) is proved.

To prove (ii), let us assume $(q_1, 1)$ is in $\sigma^d \circ F_a^W \circ W >$. There are elements w_1 and w_2 such that (q_1, w_2) is in σ^d and (w_1, w_2) is in F_a^W . Hence, there is some element q such that (w_1, q) is in $\sigma^{-d} \circ F_a$. Therefore, there is an element q_1' such that (q_1', w_1) is in σ^d and (q_1', q) is in F_a . Hence, (q_1, q_1') is in $\sigma^d \circ \sigma^{-d} = \sigma$ and since σ is a generalized congruence, it satisfies $\sigma \circ F_a \circ Q > \subseteq F_a \circ Q >$. Now, $(q_1, 1)$ is in $\sigma \circ F_a \circ Q >$ so it is in $F_a \circ Q >$, thus proving (ii).

To prove (iii), let us assume $(w_1, 1)$ is in $\sigma^{-d} \circ F_a \circ Q >$. Thus, there is at least one element q such that (w_1, q) is in $\sigma^{-d} \circ F_a$. Let us define P as the set of all such elements q . For any two elements q, q' in P there are associated elements q_1 and q_1' such that (q_1, w_1) and (q_1', w_1) are in σ^d and (q_1, q) and (q_1', q') are in F_a . Thus, (q_1, q_1') is in $\sigma^d \circ \sigma^{-d} = \sigma$ and (q, q') is in $F_a^{-1} \circ \sigma \circ F_a$. But, σ is a generalized congruence and therefore satisfies $F_a^{-1} \circ \sigma \circ F_a \subseteq \sigma$. Therefore, (q, q') is in σ and since q and q' were arbitrarily chosen from P we see that P is a complete set with respect to σ . Using Zorn's lemma as in lemma 2 of section 1.1.6, we see that P is included in some maximal complete set $K \in \mathcal{K}$. Thus, there is an element $w_2 \sim K$ such that (q, w_2) is in σ^d for each $q \in P$. But, P is just the set of all q such that (w_1, q) is in $\sigma^{-d} \circ F_a$ so we therefore have the result that (w_1, w_2) is in F_a^W . Hence, $(w_1, 1)$ is in $F_a^W \circ W >$ and (iii) is proved. This completes the proof of the theorem.

Corollary 1

If σ is a congruence relation on S , then σ^d is an epimorphism from S to S^W .

Proof: When σ is a congruence relation it is an equivalence relation on Q . Hence, by remark 4 of section 1.1.6, we see that σ^d is single valued. Since σ is not empty, the empty set is not in \mathcal{K} and hence σ^d is surjective. By theorem 8, it is therefore an epimorphism.

Corollary 2

If σ is a homomorphism from S to S' , then $(\sigma \circ \sigma^{-1})^{-d} \circ \sigma$ is a monomorphism from S^W to S' .

Proof: If σ is a homomorphism, then $\sigma \circ \sigma^{-1}$ is a congruence relation so $(\sigma \circ \sigma^{-1})^d$ is an epimorphism. Write $\theta = (\sigma \circ \sigma^{-1})^{-d} \circ \sigma$.

We wish to show that $\theta \circ \theta^{-1} = I_W$ and $\theta^{-1} \circ \theta \subseteq I_{Q'}$. Now,

$$\theta \circ \theta^{-1} = (\sigma \circ \sigma^{-1})^{-d} \circ \sigma \circ \sigma^{-1} \circ (\sigma \circ \sigma^{-1})^d = (\sigma \circ \sigma^{-1})^{-d}.$$

$$(\sigma \circ \sigma^{-1})^{-d} \circ (\sigma \circ \sigma^{-1})^d \circ (\sigma \circ \sigma^{-1})^{-d} \circ (\sigma \circ \sigma^{-1})^d = I_W \circ I_W = I_W.$$

$$\text{Also } \theta^{-1} \circ \theta = \sigma^{-1} \circ (\sigma \circ \sigma^{-1})^d \circ (\sigma \circ \sigma^{-1})^{-d} \circ \sigma = \sigma^{-1} \circ \sigma \circ \sigma^{-1}.$$

$$\sigma \subseteq I_{Q'} \circ I_{Q'} = I_{Q'}.$$

Corollary 3

If σ is an epimorphism from S to S' , then

$(\sigma \circ \sigma^{-1})^{-d} \circ \sigma$ is an isomorphism from S^W to S' .

The proof of this result is trivial. Corollaries 1, 2, and 3 are analogous to results of a similar nature concerning congruences on algebras. If A is an algebra and \equiv is a congruence, then it is customary to speak of the quotient algebra A / \equiv whose elements correspond

to congruence classes. In our notation, if S is a relational system and σ is a generalized congruence on S then S^W is the quotient system. The analogy is exact when S is a monadic algebra and σ is a congruence on S .

1.2.6 Let S and S' be two semigroups. Then a relation $\sigma \subseteq S \times S'$ will be called a generalized semigroup congruence iff whenever (s_1, s_1') is in s and (s_2, s_2') is in σ then $(s_1 s_2, s_1' s_2')$ is in σ . The analogy between this substitution property and that of the previous section should be clear. Properties (ii) and (iii) in the previous definition are not applicable in this case since the semigroup operation is single valued.

Remark 5

The property of being a generalized semigroup congruence is preserved under Pierce product and under infinite set theoretical intersection whenever these operations are applicable.

Proof: Suppose σ_1 and σ_2 are generalized semigroup congruences and $\sigma_1 \circ \sigma_2$ can be formed. To show that it is also a generalized semigroup congruence, let (s_1, s_1'') and (s_2, s_2'') be in $\sigma_1 \circ \sigma_2$. Then, we can find elements s_1' and s_2' such that (s_1, s_1') and (s_2, s_2') are in σ_1 while (s_1', s_1'') and (s_2', s_2'') are in σ_2 . Hence, $(s_1 s_2, s_1' s_2')$ is in σ_1 and (s_1', s_1'') and (s_2', s_2'') are in σ_2 . Thus, $(s_1 s_2, s_1' s_2')$ is in $\sigma_1 \circ \sigma_2$ and the first half of the remark is proved.

Next, suppose that Σ is a set of generalized semigroup congruences σ , all between S and S' . Let (s_1, s_1') and (s_2, s_2') both be in $\bigcap_{\sigma \in \Sigma} \sigma$. Then, they are in each $\sigma \in \Sigma$. Consequently, $(s_1 s_2, s_1' s_2')$ is in each $\sigma \in \Sigma$ so it is in $\bigcap_{\sigma \in \Sigma} \sigma$. This completes the proof of the remark.

Remark 6

If σ is a reflexive generalized semigroup congruence on a single semigroup S , then $\sigma^+ = \sigma^*$ is also.

Proof: This remark could be proved easily if we had closure under set theoretical union as in the case of relational systems. However, such closure does not hold in the semigroup case. We therefore assume (s_1, s_1') and (s_2, s_2') are both in σ^+ . Hence, for some m and n , we have $(s_1, s_1') \in \sigma^m$ and $(s_2, s_2') \in \sigma^n$. Therefore $(s_1 s_2, s_1' s_2')$ is in $\sigma^m \circ \sigma^n = \sigma^{m+n} \subseteq \sigma^+$, and the remark is proved.

Using the same defining rules as we used for generalized congruences on relational systems, we can define such concepts as semigroup congruences and semigroup homomorphisms, the latter being further classified as epimorphisms, endomorphisms, etc.

Remark 7

The mapping F of a relational system $S = (A, Q, F)$, when extended to form a mapping from A^* to Q^* is a semigroup homomorphism from the free monoid generated by A to the input monoid of S .

Proof: The substitution property is just our previously noted rule:

$F_{st} = F_s \circ F_t$ for all s, t in A^* . Hence, this result is obvious.

1.2.7 Let Q be any set and Ω an operator mapping $\mathcal{P}(Q)$ either to itself or another family of sets. Then, we shall call Ω a distributive operator iff for every family \mathcal{J} of subsets T of Q the equation

$$\Omega\left(\bigcup_{T \in \mathcal{J}} T\right) = \bigcup_{T \in \mathcal{J}} \Omega(T)$$

holds. A special case of the above equation occurs when \mathcal{J} is empty, giving $\Omega(\phi) = \phi$. The property of finite distributivity requires only that the defining equation hold for finite nonempty families \mathcal{J} .

Remark 8

A necessary and sufficient condition for Ω to be distributive is that $\Omega(T) = \bigcup_{q \in T} \Omega(q)$ for every subset T of Q .

Proof: This condition is clearly necessary since T is the union of singleton sets of its elements. It is also sufficient since

$$\Omega\left(\bigcup_{T \in \mathcal{J}} T\right) = \Omega\left(\bigcup_{q \in \bigcup_{T \in \mathcal{J}} T} \{q\}\right) = \bigcup_{T \in \mathcal{J}} \Omega\left(\bigcup_{q \in T} \{q\}\right) = \bigcup_{T \in \mathcal{J}} \Omega(T).$$

Lemma 3

If Ω is a distributive operator mapping $\mathcal{P}(Q)$ to itself and if $\Omega^*(T)$ denotes the closure of Ω applied to the subset T of Q , then $\Omega^*(T)$ is the minimum of all subsets T' of Q such that $T' \supseteq T$ and $T' \supseteq \Omega(T')$.

Proof: Abbreviate $\Omega(\Omega(\dots(T)\dots))$ as $\Omega^n(T)$, where n is the number of times Ω is applied. Then, we may write $\Omega^*(T) = \bigcup_{n=0}^{\infty} \Omega^n(T)$, where $\Omega^0(T) = T$. Thus, we see that $\Omega^*(T) \supseteq T$ and $\Omega^*(T) \supseteq \Omega(\Omega^*(T))$, so $\Omega^*(T)$ satisfies the two given conditions. To show that it is a minimum, we take T' to be any set such that $T' \supseteq T$ and $T' \supseteq \Omega(T')$. Let q be an element of $\Omega^*(T)$. Then q is in $\Omega^n(T)$ for some $n \geq 0$. Hence, for some sequence $q_0, q_1, \dots, q_n = q$, we have $q_0 \in T$ and $q_i \in \Omega(q_{i-1})$ for $i=1, \dots, n$ by remark 8. Thus, by induction, we see that q_0 is in T' and each q_i is in T' since q_{i-1} is in T' . Hence, each term in the sequence is in T' , and q is therefore in T' . This completes the proof since we have shown that $\Omega^*(T)$ is a subset of any T' .

Remark 9

Given a pair S, S' of similar relational systems, not necessarily distinct, and a relation $\sigma_0 \subseteq Q \times Q'$, then there is a unique minimum relation $\sigma \subseteq Q \times Q'$ satisfying (i) $\sigma \supseteq \sigma_0$,

and (ii) $\sigma \supseteq \bigcup_{a \in A} F_a^{-1} \circ \sigma \circ F_a'$.

If we define $\Omega(X) = \bigcup_{a \in A} F_a^{-1} \circ X \circ F_a'$, then $\sigma = \Omega^*(\sigma_0)$. Also, if there is any generalized congruence σ' between S and S' such that $\sigma' \supseteq \sigma_0$, then σ is the minimum such generalized congruence.

Proof: The first part of this remark follows from lemma 3 and the observation that the operator Ω of the remark is distributive. The second part of the remark can be proved by first noting that

$\sigma \subseteq \sigma'$ since σ is minimal over relations satisfying (i) and (ii). Therefore, $\sigma \circ F_a' \circ Q' > \subseteq \sigma' \circ F_a' \circ Q' > \subseteq F_a \circ Q > .$ Similarly, we show that $\sigma^{-1} \circ F_a \circ Q > \subseteq F_a' \circ Q' > .$ Thus, σ is a generalized congruence and the proof is complete.

We may think of remark 9 as providing us with an algorithm for constructing $\sigma = \Omega^*(\sigma_0)$ when σ_0 is given. To carry out this algorithm, we define $\sigma_{i+1} = \sigma_i \cup \Omega(\sigma_i) = \sigma_i \cup \bigcup_{a \in A} F_a^{-1} \circ \sigma_i \circ F_a'$, for $i=0, 1, 2, \dots$. Then each element of σ is contained in some σ_i . While this algorithm appears to be infinite, there are many important cases in which it may be terminated after some finite number n of steps. In particular, if $\sigma_{n+1} = \sigma_n$, then clearly $\sigma_n = \sigma$. It is also clear that when each F_a and each F_a' is finitary, the algorithm terminates. A fairly obvious special case of this situation occurs when the relational systems S and S' are finite, i. e. Q and Q' are finite.

Theorem 9

Let $S = (A, Q, F)$ and $S' = (A, Q', F')$ be similar relational systems and let $\sigma_0 \subseteq Q \times Q'$ be a relation such that for all $a \in A$, $\sigma_0 \circ F_a' \circ Q' > \subseteq F_a \circ Q >$ and $\sigma_0^{-1} \circ F_a \circ Q > \subseteq F_a' \circ Q' > .$ Define $\Omega(X) = \bigcup_{a \in A} F_a \circ X \circ (F_a')^{-1}$. Then $\sigma = \overline{\Omega^*(\sigma_0)}$ is the unique maximum generalized congruence relation between S and S' such that $\sigma \subseteq \sigma_0$. If $S = S'$ and σ_0 is symmetric, then σ is symmetric. If $S = S'$ is single valued and σ_0 is an equivalence relation, then σ is a congruence relation.

Proof: The operator Ω as defined in the theorem is clearly a distributive operator. By lemma 3, we see that $\bar{\sigma}$ is the minimum relation satisfying $\bar{\sigma} \supseteq \bar{\sigma}_0$ and $F_a \circ \bar{\sigma} \circ (F_a')^{-1} \subseteq \bar{\sigma}$ for each $a \in A$. The condition $\bar{\sigma} \supseteq \bar{\sigma}_0$ may be rewritten $\sigma \subseteq \sigma_0$. Hence, $\sigma \circ F_a' \circ Q' > \subseteq \sigma_0 F_a' \circ Q' > \subseteq F_a \circ Q >$. Also, $\sigma^{-1} \subseteq \sigma_0^{-1}$, so $\sigma^{-1} \circ F_a \circ Q > \subseteq \sigma_0^{-1} \circ F_a \circ Q > \subseteq F_a' \circ Q' >$. The condition $F_a \circ \bar{\sigma} \circ (F_a')^{-1} \subseteq \bar{\sigma}$ is equivalent to $F_a^{-1} \circ \sigma \circ F_a' \subseteq \sigma$, by the following argument:

$$F_a \circ \bar{\sigma} \circ (F_a')^{-1} \subseteq \bar{\sigma} \iff \text{Tr}(F_a \circ \bar{\sigma} \circ (F_a')^{-1} \circ \sigma^{-1}) = 0 \iff$$

$$\text{Tr}(\sigma \circ F_a' \circ \bar{\sigma}^{-1} \circ F_a^{-1}) = 0 \iff \text{Tr}(F_a^{-1} \circ \sigma \circ F_a' \circ \bar{\sigma}^{-1}) = 0 \iff$$

$$F_a^{-1} \circ \sigma \circ F_a' \subseteq \sigma. \text{ Thus, the two conditions are equivalent to } \sigma \text{ being}$$

a generalized congruence such that $\sigma \subseteq \sigma_0$. Since $\bar{\sigma}$ is a minimum, we see that σ is the maximum such generalized congruence.

Next, we assume that $S = S'$ and that σ_0 is symmetric. Hence, $\bar{\sigma}_0$ is also symmetric. Define recursively, $\bar{\sigma}_{i+1} = \bar{\sigma}_i \cup \Omega(\bar{\sigma}_i)$, for $i=0, 1, \dots$. Taking $\bar{\sigma}_i$ as symmetric, so is $\Omega(\bar{\sigma}_i)$ from its definition, so $\bar{\sigma}_{i+1}$ is symmetric, completing the inductive step. But, if (q, q') is in $\bar{\sigma}$ then (q, q') is in $\bar{\sigma}_i$ for some i and hence (q', q) is in $\bar{\sigma}_i \subseteq \bar{\sigma}$. Thus, $\bar{\sigma}$ is symmetric so σ is symmetric.

Finally, we assume that $S = S'$ is single valued and that σ_0 is an equivalence relation. Thus, $\bar{\sigma}_0$ is irreflexive. Let us assume inductively that $\bar{\sigma}_i$ is irreflexive. Then $F_a \circ \bar{\sigma}_i \circ F_a^{-1}$ is irreflexive because F_a is determinate. Hence, $\Omega(\bar{\sigma}_i)$ is irreflexive and so is $\bar{\sigma}_{i+1}$. Since each element of $\bar{\sigma}$ is in some $\bar{\sigma}_i$, we see that $\bar{\sigma}$ is irreflexive and thus σ is reflexive. Now σ_0 is transitive so $\sigma_0 \circ \sigma_0 \subseteq \sigma_0$.

Because of the symmetry of σ_0 , this property is equivalent to

$$\begin{aligned} \sigma_0 \circ \sigma_0 &\subseteq \sigma_0 \text{ by the following argument: } \sigma_0 \circ \sigma_0 \subseteq \sigma_0 \iff \\ \text{Tr}(\sigma_0 \circ \sigma_0 \circ \bar{\sigma}_0^{-1}) &= 0 \iff \text{Tr}(\bar{\sigma}_0 \circ \sigma_0^{-1} \circ \sigma_0^{-1}) = 0 \iff \\ \text{Tr}(\sigma_0^{-1} \circ \bar{\sigma}_0 \circ \sigma_0^{-1}) &= 0 \iff \sigma_0^{-1} \circ \bar{\sigma}_0 \subseteq \bar{\sigma}_0 \iff \sigma_0 \circ \bar{\sigma}_0 \subseteq \bar{\sigma}_0. \end{aligned}$$

Let us assume inductively that σ_i is transitive. We wish to prove

that σ_{i+1} is transitive so we write $\sigma_{i+1} \circ \bar{\sigma}_{i+1} = \sigma_{i+1}$.

$$(\bar{\sigma}_i \cup \bigcup_{a \in A} F_a \circ \bar{\sigma}_i \circ F_a^{-1}) = \sigma_{i+1} \circ \bar{\sigma}_i \cup \bigcup_{a \in A} \sigma_{i+1} \circ F_a \circ \bar{\sigma}_i \circ F_a^{-1}.$$

Now, $\sigma_{i+1} \subseteq \sigma_i$, so $\sigma_{i+1} \circ \bar{\sigma}_i \subseteq \sigma_i \circ \bar{\sigma}_i \subseteq \bar{\sigma}_i$ by the transitivity of σ_i .

Also, $F_a \circ \bar{\sigma}_i \circ F_a^{-1} \subseteq \bar{\sigma}_{i+1}$ by the definition of $\bar{\sigma}_{i+1}$ and hence

$$\text{Tr}(F_a \circ \bar{\sigma}_i \circ F_a^{-1} \circ \sigma_{i+1}) = 0 \text{ giving } \text{Tr}(F_a^{-1} \circ \sigma_{i+1} \circ F_a \circ \bar{\sigma}_i) = 0$$

$$\iff F_a^{-1} \circ \sigma_{i+1} \circ F_a \subseteq \sigma_i. \text{ But } F_a \text{ is productive, so } I_Q \subseteq F_a \circ F_a^{-1}.$$

Hence $\sigma_{i+1} \circ F_a \subseteq F_a \circ F_a^{-1} \circ \sigma_{i+1} \circ F_a \subseteq F_a \circ \sigma_i$. Thus,

$$\bigcup_{a \in A} \sigma_{i+1} \circ F_a \circ \bar{\sigma}_i \circ F_a^{-1} \subseteq \bigcup_{a \in A} F_a \circ \sigma_i \circ \bar{\sigma}_i \circ F_a^{-1} \subseteq \bigcup_{a \in A} F_a \circ \bar{\sigma}_i \circ F_a^{-1}.$$

This gives us $\sigma_{i+1} \circ \bar{\sigma}_{i+1} \subseteq \bar{\sigma}_i \cup \bigcup_{a \in A} F_a \circ \bar{\sigma}_i \circ F_a^{-1} = \bar{\sigma}_{i+1}$. Hence,

σ_{i+1} is transitive completing our inductive step and the proof of the theorem.

As was the case with our previous result, remark 9, this theorem may also be regarded as defining an algorithm for finding a congruence relation σ with the properties described in the theorem. It is true that the algorithm may not always be finite, but in many practical cases it may be shown to terminate after a finite number of steps. We shall call this the Moore algorithm since the process was invented by E. F. Moore for the case in which S is a finite monadic algebra and σ_0 is an

equivalence relation. In this case σ is a congruence relation and the process terminates after a finite number of steps. Moore was able to construct σ^d which is an epimorphism and S^W which is the image system under σ^d . He called S^W the "reduced machine."

1. 2. 8 A partial ordering on a set Q was defined in section 1. 1. 3 as a relation $\sigma \subseteq Q \times Q$ which is reflexive, antisymmetric, and transitive. One example of a partial ordering is the relation \subseteq of set inclusion among a family \mathcal{J} of subsets of a set T . We see that the relation $T_i \subseteq T_j$ between two such subsets must be reflexive, antisymmetric, and transitive. Therefore, it is a partial order. This means that any family of relations on the same pair $Q \times Q'$ of sets is partially ordered by set inclusion. However, other examples of partially ordered sets abound in mathematics and in everyday experience.

If $\sigma \subseteq Q \times Q$ is a partial ordering and P is a subset of Q , then we say that q is an upper bound of P iff (p, q) is in σ whenever p is in P . Similarly, we say q' is a lower bound of P iff (q', p) is in σ whenever p is in P . A set P may or may not possess upper bounds and lower bounds when σ is any given partial ordering. However, an important special case occurs when every nonempty set P not only has an upper bound and a lower bound but it has a unique minimum upper bound and a unique maximum lower bound. A partial ordering of this type is called a complete lattice. Specifically, there is a unique upper bound, called the supremum of P and written $\text{Sup}(P)$, which has the property that if q is any upper bound of P , then $(\text{Sup}(P), q)$ is in σ .

Also, there is a unique lower bound, called the infimum of P and written $\text{Inf}(P)$, which has the property that if q' is any lower bound of P , then $(q', \text{Inf}(P))$ is in σ .

In case it is only possible to assert that $\text{Sup}(P)$ and $\text{Inf}(P)$ exist when P is finite, we say simply that σ is a lattice, the adjective "complete" being reserved for lattices in which it is known to be true in general.

A notation that is frequently used with lattices is to write $p \cup q$ and $p \cap q$ to represent the supremum and infimum respectively of the two element set $\{p, q\}$. A great disadvantage of this notation is that the same symbols are used for set theoretical union and intersection. When we discuss the partial ordering among sets in a family this notation is clearly ambiguous because the set theoretical and lattice theoretical operations may not signify the same result. It will be necessary to avoid such ambiguity by carefully explaining which type of operation is intended in cases of uncertainty.

A great convenience of the $p \cup q$ and $p \cap q$ notation is that one may derive simple algebraic rules which apply to lattices. The reader may verify the following:

1. $p \cap (q \cap r) = (p \cap q) \cap r$ and $p \cup (q \cup r) = (p \cup q) \cup r$, associativity.
2. $p \cap q = q \cap p$ and $p \cup q = q \cup p$, commutativity.
3. $p \cap p = p$ and $p \cup p = p$, idempotence.
4. $p \cap (p \cup q) = p$ and $p \cup (p \cap q) = p$, absorption.

Many other identities may be derived. It is also possible to regard the operations $p \cup q$ and $p \cap q$ as fundamental and from their properties derive the properties of partial ordering in a lattice. For this purpose we may define σ by either of the two rules:

$$5. \quad p \cap q = p \iff (p, q) \in \sigma, \text{ or}$$

$$5'. \quad p \cup q = q \iff (p, q) \in \sigma, \text{ both of which are equivalent.}$$

By introducing this algebraic notation we have changed our emphasis from the set σ to the set Q . From the point of view of the operations \cup and \cap , a lattice is an algebra whose carrier set is Q .

When the entire carrier set Q of a lattice has a supremum and an infimum these elements are written I and O respectively. We note that $\text{Sup}(Q) = I$ and $\text{Inf}(Q) = O$ always exist in the case of complete lattices. These elements obey the special rules:

$$6. \quad O \cap p = O \text{ and } I \cup p = I.$$

$$7. \quad O \cup p = p \text{ and } I \cap p = p.$$

If \mathcal{J} is a family of subsets of a set T and \mathcal{J} forms a lattice under set inclusion, then the relation $T_i \subseteq T_j$ between members of \mathcal{J} has the same meaning in the lattice \mathcal{J} and in the family of sets \mathcal{J} . However, the set theoretical union and intersection of a subfamily \mathcal{J}' of \mathcal{J} , which we shall write as $\bigcup_{T' \in \mathcal{J}'} T'$ and $\bigcap_{T' \in \mathcal{J}'} T'$ respectively may not be the same as the lattice theoretical $\text{Sup}(\mathcal{J}')$ and $\text{Inf}(\mathcal{J}')$.

Remark 10

We always have $\bigcup_{T' \in \mathcal{J}'} T' \subseteq \text{Sup}(\mathcal{J}')$ and $\text{Inf}(\mathcal{J}') \subseteq \bigcap_{T' \in \mathcal{J}'} T'$, if $\text{Sup}(\mathcal{J}')$ and $\text{Inf}(\mathcal{J}')$ exist. However, when $\bigcup_{T' \in \mathcal{J}'} T'$ is in \mathcal{J} then it is $\text{Sup}(\mathcal{J}')$, and when $\bigcap_{T' \in \mathcal{J}'} T'$ is in \mathcal{J} then it is $\text{Inf}(\mathcal{J}')$.

Proof: Let T' be any element of \mathcal{J}' . Then $T' \subseteq \text{Sup}(\mathcal{J}')$ by the definition of $\text{Sup}(\mathcal{J}')$. Hence, taking the union over \mathcal{J}' , we obtain $\bigcup_{T' \in \mathcal{J}'} T' \subseteq \text{Sup}(\mathcal{J}')$. If $\bigcup_{T' \in \mathcal{J}'} T'$ is in \mathcal{J} we see that it is an upper bound of \mathcal{J}' . Hence, $\bigcup_{T' \in \mathcal{J}'} T' \supseteq \text{Sup}(\mathcal{J}')$ by the definition of $\text{Sup}(\mathcal{J}')$. This yields $\bigcup_{T' \in \mathcal{J}'} T' = \text{Sup}(\mathcal{J}')$ and one half of the remark is proved. The second half may be proved by a dual argument.

Remark 11

The family of all generalized congruences σ between two relational systems S and S' forms a complete lattice under set inclusion.

Proof: Since this set of generalized congruences is closed under set theoretical union and intersection, we may use remark 10 to show that the supremum and infimum of any family of generalized congruences between S and S' must be just the union and intersection respectively. Thus, the proof follows.

The following theorem is a generalization of a result due to E. H. Moore (not E. F. Moore) and is useful in determining which partially ordered sets are lattices.

Theorem 10

If $\sigma \subseteq Q \times Q$ is a partial ordering of Q such that every nonempty subset P of Q has an infimum and Q has an upper bound I , then σ is a complete lattice.

Proof: To prove this theorem, we must show that every nonempty subset P of Q has a supremum. Let P' be the set of upper bounds of P . We see that P' is nonempty since I is an upper bound of P and therefore a member of P' . Hence, $\text{Inf}(P')$ exists and by its definition, we see that $(q, \text{Inf}(P'))$ is in σ whenever q is in P so $\text{Inf}(P')$ is in P' . Also, $(\text{Inf}(P'), q')$ is in σ whenever q' is in P' , so $\text{Inf}(P') = \text{Sup}(P)$. This completes the proof of the theorem since P was chosen arbitrarily.

It may be noted that the dual of theorem 10 may be stated and is equally valid.

Given two partial orderings $\sigma_1 \subseteq Q_1 \times Q_1$ and $\sigma_2 \subseteq Q_2 \times Q_2$ such that $Q_1 \subseteq Q_2$ and $\sigma_1 \subseteq \sigma_2$, we say that σ_1 is a subordering of σ_2 . However, if σ_1 and σ_2 are both lattices we would not generally assert that σ_1 is a sublattice of σ_2 because we regard a lattice as an algebra. To illustrate this difference, let $p \underset{1}{\vee} q$ and $p \underset{2}{\vee} q$ represent the supremum of $\{p, q\}$ in σ_1 and σ_2 respectively. Clearly, $(p \underset{2}{\vee} q, p \underset{1}{\vee} q)$ is in σ_2 , but we cannot generally conclude that $p \underset{1}{\vee} q$

and $p \cup_2 q$ are equal. The same comment applies to the infimum. Thus, we shall only say that σ_1 is a sublattice of σ_2 when the supremum and infimum of any set in the first system are the same as the supremum and infimum of that set in the second.

The following two theorems are analogous to results obtained by Garrett Birkhoff concerning congruence relations on algebras

Theorem 11

The family \sim of equivalence relations $\theta \subseteq Q \times Q$ on a set Q forms a complete lattice under set inclusion. If \mathcal{C} is a nonempty subset of \sim , then $\text{Inf}(\mathcal{C}) = \bigcap_{\theta \in \mathcal{C}} \theta$ and $\text{Sup}(\mathcal{C}) = (\bigcup_{\theta \in \mathcal{C}} \theta)^*$.

Proof: We note that if \mathcal{C} is a nonempty set of equivalence relations, then $\bigcap_{\theta \in \mathcal{C}} \theta$ is an equivalence relation. Therefore,

$\bigcap_{\theta \in \mathcal{C}} \theta = \text{Inf}(\mathcal{C})$. Furthermore, $Q \times Q$ is an equivalence relation

so \sim has an upper bound. Hence, by theorem 10, \sim is a complete lattice under set inclusion. It is also not difficult to see that $\text{Sup}(\mathcal{C}) = (\bigcup_{\theta \in \mathcal{C}} \theta)^*$ because $\text{Sup}(\mathcal{C}) \supseteq \bigcup_{\theta \in \mathcal{C}} \theta$. Since $\bigcup_{\theta \in \mathcal{C}} \theta)^*$

is reflexive and symmetric and is the minimum transitive relation including $\bigcup_{\theta \in \mathcal{C}} \theta$, it is the minimum equivalence relation including

$\bigcup_{\theta \in \mathcal{C}} \theta$. Thus the theorem is proved.

Theorem 12

The family Σ of congruence relations $\sigma \subseteq Q \times Q$ on a relational system $S = (A, Q, F)$ forms a complete lattice under set inclusion which is a sublattice of the lattice of equivalence relations on Q .

Proof: Assume that Σ is nonempty so that the lattice of congruence relations if it exists is nondegenerate. Then, if \mathcal{C} is a nonempty family of congruence relations σ on S , we see that $\bigcap_{\sigma \in \mathcal{C}} \sigma$ is a congruence relation because it is both an equivalence relation and a generalized congruence. Also, $(\bigcup_{\sigma \in \mathcal{C}} \sigma)^* = (\bigcup_{\sigma \in \mathcal{C}} \sigma)^+$ since each $\sigma \in \mathcal{C}$ is reflexive. From our results concerning closure of generalized congruences under union and transitive extension, we see that $(\bigcup_{\sigma \in \mathcal{C}} \sigma)^+$ is a generalized congruence and since it is an equivalence, it must be a congruence. Hence, both $\text{Sup}(\mathcal{C})$ and $\text{Inf}(\mathcal{C})$ are in Σ , so it forms a sublattice of the lattice of equivalence relations, and the theorem is proved.

It is interesting to compare theorem 12 with remark 11. From remark 11 we see that the family of generalized congruences on a system S forms a complete lattice under set inclusion. The subset of this lattice consisting of all congruences is also a complete lattice under set inclusion. However, the second lattice is not generally a sublattice of the first. It is clear that the reason is that the union of a set of equivalence relations is not generally transitive and hence not another equivalence relation. Therefore the $\text{Sup}(\mathcal{C})$ over the lattice of generalized

congruences may differ from the $\text{Sup}(\mathcal{C})$ over the lattice of congruences.

Section 2

Use of Multiple Index Matrices in Generalized Automata Theory

2.1 Introduction

An important generalization of automata theory occurs when one allows the transition function of an automaton to have several arguments. An ordinary automaton has a function f_α associated with each input configuration α which maps the present state i to the next state j . However, in the generalization, these functions may have several arguments $i(1), \dots, i(p)$ rather than a single argument i . Mezei and Wright [31], and Thatcher and Wright [42] have shown that many of the results of automata theory carry over in this generalization.

It is no longer reasonable to regard the arguments $i(1), \dots, i(p)$ as states of a machine, since we may think of a machine as possessing only a single state at a given time. Nevertheless, it is likely that the new automata theory will have greater and more direct applicability to machines than the old, because the operations actually carried out by computers do usually have several arguments, and it is much more natural and convenient to describe a computer in terms of the operations it can perform than by means of a state transition table or diagram.

Difficulties with this generalization have not been so much conceptual as notational. Thus, with the object of placing the entire theory within the framework of modern mathematics, Eilenberg and Wright [7] have represented the generalized automata as categories, drawing heavily upon earlier work of Lawvere [26] in universal algebra.

In this paper we wish to describe an alternative method of systematizing the treatment of generalized automata in a way which we believe presents a number of advantages :

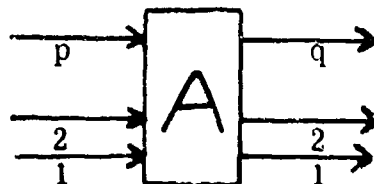
- (1) The method is based on familiar mathematical notions which are used frequently in switching and automata theory. It does not require the introduction of category theory although category theory may be applied to this as to many other mathematical systems.
- (2) There is a convenient way of visualizing the constructions of the system which enables one to anticipate results.
- (3) The present approach is in some respects similar to the product and permutation category (PROP) method described by MacLane [29], but is based upon a single operation and permits the derivation of such notions as direct product and its properties in terms of this operation. Also, there is complete left-right duality in the theory.
- (4) There is a natural generalization to relational systems which does not require the assumption of a distributive law with respect to set union. A further possibility of generalization has to do with matrices of numbers.

2.2 Multiple Index Matrices

If A is a matrix, then a_{ij} is often written to represent the element in the i -th row and the j -th column of A . However, to avoid cumbersome subscripts, we shall write $[i(1) \dots i(p) \ aj(1) \dots j(q)]$ to represent the matrix element when there are p row indices $i(1), \dots, i(p)$ and q column indices $j(1), \dots, j(q)$. Our concern will be with matrices in which all indices have the same range K , where K is assumed to contain at least two elements. The matrix elements are the Boolean quantities 1 or 0, i.e. truth or falsity, but most of what we have to say applies equally well to matrices whose elements are numbers.

We shall call the above matrix A a (p, q) index matrix and allow p and q to be any integers greater than or equal to zero. When $p = 0$ and $q > 0$ we call A a row vector and when $p > 0$ and $q = 0$ we call it a column vector. If $p = q = 0$, it is called a scalar.

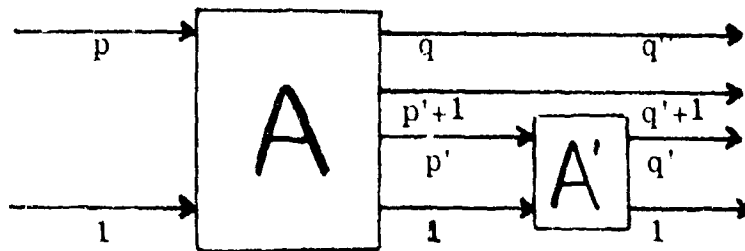
An alternative terminology derived from network theory is to call p and q the numbers of input and output lines respectively. We can thus represent A as a box in the diagram below.



We interpret this diagram as a device or process A which requires p signals or arguments, taken from K and denoted by $i(1), \dots, i(p)$, on the input lines $1, \dots, p$ respectively. It produces some configuration of q signals or results, also taken from K and denoted by $j(1), \dots, j(q)$, on the corresponding output lines, such that the matrix element $[i(1) \dots i(p) \ a_j(1) \dots j(q)]$ has the value 1, i. e. is a true proposition. The sequence $j(1), \dots, j(q)$ may not be precisely specified by this requirement just as the "next state" of a nondeterministic machine is not precisely specified and may be regarded as depending upon unspecified data or conditions.

Our basic rule concerns the formation of matrix product. Let A and A' be (p, q) index and (p', q') index matrices respectively. Then the matrix product $AA' = B$ is defined by treating two cases.

Case 1. If $q \geq p'$, then B has p row indices and $q'' = q' + p - p'$ column indices. Each element of B is defined by the rule: $[i(1) \dots i(p) \ b_j(1) \dots j(q'')]$ $= \sum [i(1) \dots i(p) \ a_k(1) \dots k(p') \ j(q'+1) \dots j(q'')] [k(1) \dots k(p') \ a'_j(1) \dots j(q')]$ $k(1) \dots k(p') \in K$. A diagram representing this case is shown below.

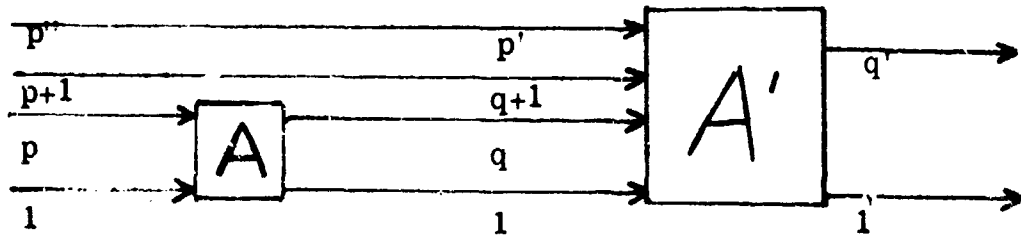


Case 2. If $q \leq p'$, then B has $p'' = p + p' - q$ row indices and q' column indices. Each element of B is defined by the rule: $[i(1) \dots$

$$i(p'') \ b_j(1) \dots j(q')] = \Sigma [i(1) \dots i(p) \ a_k(1) \dots k(q)] [k(1) \dots$$

$$k(q) \ i(p+1) \dots i(p'') \ a'_{j(1) \dots j(q')}] \ k(1) \dots k(q) \in K.$$

In this case we have the following diagram.



When working with Boolean matrices, we interpret sum and product as union and intersection in the above expression. When using strict logical notation if one regards each matrix element as a proposition, one should replace the symbol Σ by \exists , the symbol $=$ by $\langle == \rangle$, and insert $\&$ between the two elements on the right.

2.3 Linked Lines and Reduced Matrices

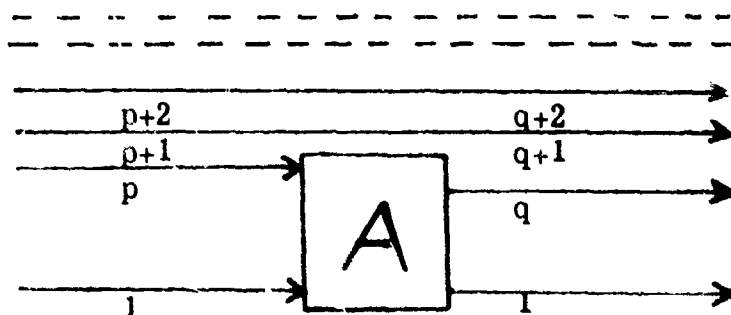
Let A be a (p, q) index matrix. Then, we shall say that the m -th row index is linked with the n -th column index if and only if

- (1) $[i(1) \dots i(p) \ a_j(1) \dots j(q)] = 0$ whenever $i(m) \neq j(n)$, and
- (2) $[i(1) \dots i(p) \ a_j(1) \dots j(q)]$ is independent of $x = i(m) = j(n)$ whenever $i(m) = j(n)$.

If the p -th row index and the q -th column index are linked, then we may reduce A to a $(p-1, q-1)$ index matrix A_1 in which the element $[i(1) \dots i(p-1) a_{ij}(1) \dots j(q-1)]$ is simply taken as equal to the element $[i(1) \dots i(p-1)x a_{ij}(1) \dots j(q-1)x]$ which is the same for all x in K . A reduced matrix is one in which the p -th row index and q -th column index are not linked. It is clear that to any matrix A there corresponds a unique reduced matrix A_r obtained by repeatedly eliminating the highest numbered row and column indices until this is no longer possible.

Two matrices may be regarded as equivalent if they have the same reduced matrix. It is possible to show that this equivalence relation is actually a congruence relation with respect to matrix multiplication. That is to say, we obtain equivalent results regardless of whether we multiply two matrices or their reductions. Thus, in the remainder of this abstract we shall work with these congruence classes of matrices, generally using reduced matrices as representatives of their classes. In particular, we shall take the numbers p, q associated with any class to be those of the reduced representative of that class.

A diagram corresponding to a congruence class of matrices may be drawn as follows.



The box labelled A corresponds to the reduced representative of the class. Above this box we imagine an infinite number of lines corresponding to the linked indices of other members of the class. Any input signal or argument on a line numbered $n > p$ will pass unchanged as a result on the linked output line numbered $n - p + q$.

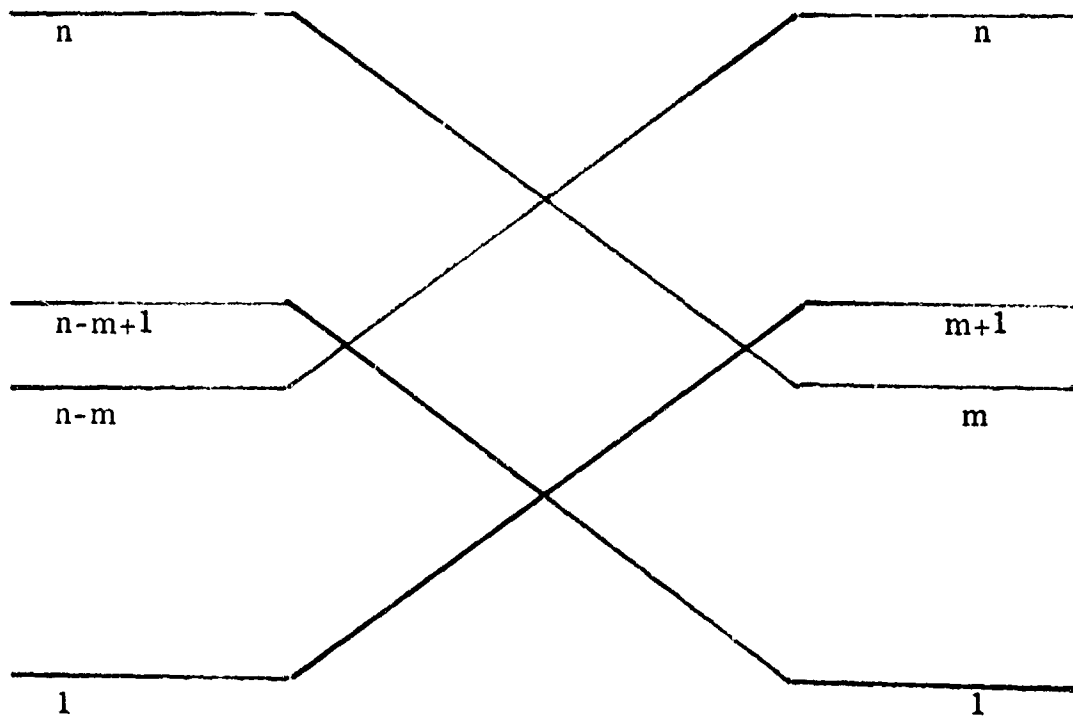
2.4 The Group

Let \mathcal{A} be the group of all finite permutations of the set of all natural numbers $\{1, 2, \dots\}$. We can construct a set \mathcal{H} of multiple index matrices which is isomorphic to \mathcal{A} with matrix multiplication corresponding to the group operation. This construction is possible for any given index range K of cardinality two or greater.

Let S be a permutation in \mathcal{A} which leaves all numbers greater than n fixed and maps $\{1, 2, \dots, n\}$ to $\{s_1, s_2, \dots, s_n\}$. We construct a corresponding matrix H with $p = q = n$ whose matrix elements $[i(1) \dots i(n) \ h_i(s_1) \dots i(s_n)]$ are 1 for any sequence $i(1), \dots, i(n)$ in K , and whose other matrix elements are all 0.

It may be shown that each row index number m of H is linked to the column index numbered s_m . Thus, on our diagram H is an operation corresponding to a scrambling of the lines when passing from input to output as given by the permutation S . It is possible to prove that matrix multiplication of members of \mathcal{H} corresponds to the product of the permutations. Note that the members of \mathcal{H} are not permutation matrices in the usual sense since they do not permute the members of K but rather the indices.

As a convenient set of generators for \mathcal{H} we shall choose the matrices π_n corresponding to the cyclic permutations $(12 \dots n)$ of the first n natural numbers. We shall write π_n^m for the m -th power of π_n . A diagram for π_n^m may be drawn as follows.



2.5 Relational Systems

The importance of the group \mathcal{H} in connection with multiplication of multiple index matrices should be clear since by choosing a suitable member H of \mathcal{H} we may make any connection AHB of output lines of A with input lines of B .

Define a relational system \mathcal{R} as any set of multiple index matrices with some common index range K , called the carrier and with the following two properties:

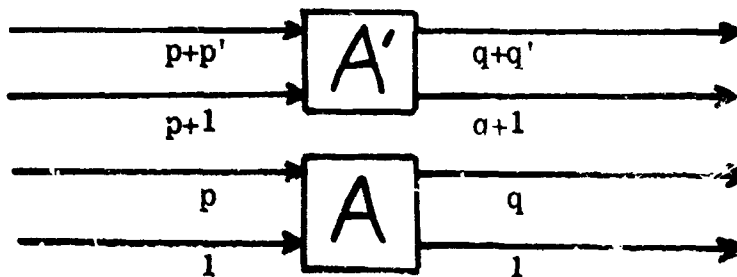
(1) \mathcal{R} is closed under matrix multiplication

(2) \mathcal{K} is a subset of \mathcal{R} .

It is possible to show that matrix multiplication is associative so \mathcal{R} is a semigroup. Also, \mathcal{K} is a group whose unit element is the scalar 1 which is a unit element for \mathcal{R} . Thus, \mathcal{R} is a monoid or semigroup with a unit.

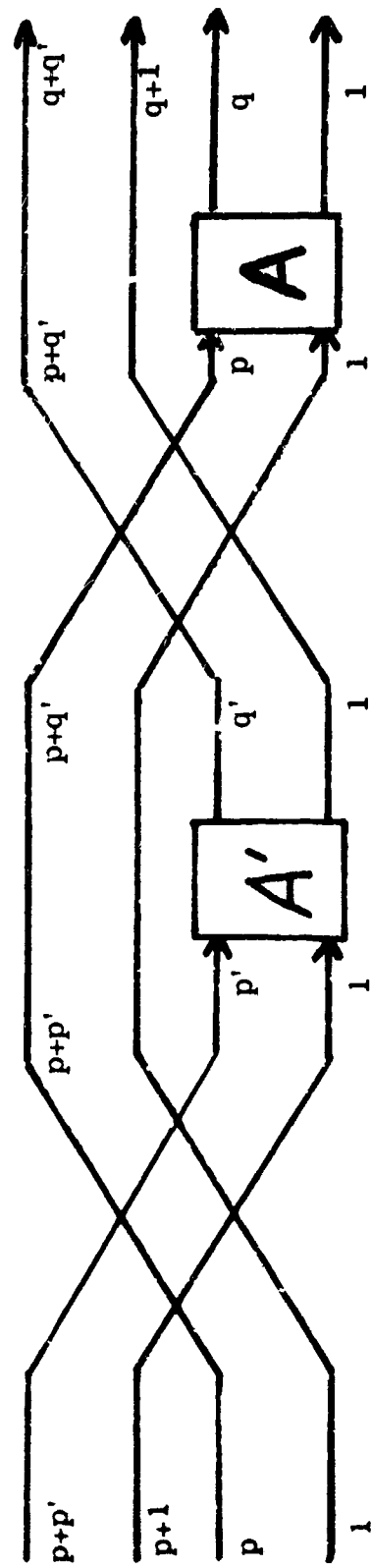
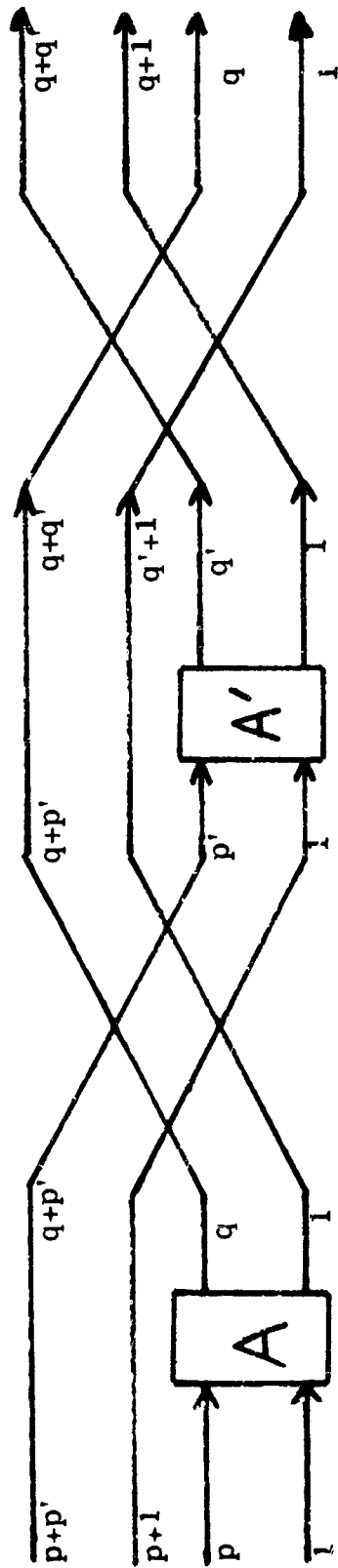
Using the definitions given earlier we may prove the commutation law: $A \pi_{q+p}^{p'} A' \pi_{q+q'}^q = \pi_{p+p}^{p'} A' \pi_{p+q}^p A$. A pictorial representation of the left and right sides of this equation is shown on the following page.

The direct product $A \otimes A'$ is defined to be the quantity appearing on either side of this equation. An alternative way of representing $A \otimes A'$ is by the diagram shown below.



It is possible to prove that direct product is associative and that it has various other properties.

An important special case of a relational system is that of an algebra. Each matrix R in \mathcal{R} is then single valued in the sense that for every sequence $i(1), \dots, i(p)$ in K , there is exactly one sequence



$j(1), \dots, j(q)$ in K such that $[i(1) \dots i(p) rj(1) \dots j(q)] = 1$. All other matrix elements have the value 0. Furthermore, \mathcal{R} can be generated from \mathcal{H} and a set \mathcal{R}_0 of matrices with $q=1$.

For example, a semigroup may be generated from a single binary operation. Letting B denote this operation, we see that $p=2$ and $q=1$. The associative law may be written: $BB = \pi_3^2 B \pi_2 B$. To describe a monoid we include the unit element E as an operation with $p=0$ and $q=1$. The unit rules are $EB = E\pi_2 B = e$. Here, e denotes the unit of \mathcal{R} rather than the unit of the monoid being described.

2.6 Relational Semigroups

A relational semigroup (RSG) is a slightly more abstract entity than a relational system. It is defined as any semigroup \mathcal{J} with the following three properties.

- (1) \mathcal{J} contains the subgroup \mathcal{A} consisting of all finite permutations of the natural numbers. Also, the unit e of \mathcal{A} is the unit of \mathcal{J} .
- (2) Associated with each member T of \mathcal{J} there are two integers $p(T)$, $q(T)$, both ≥ 0 , which obey the following rules.
 - (a) $p(TT') \leq \max(p(T), p(T') - q(T) + p(T))$
 - (b) $q(TT') \leq \max(q(T'), q(T) - p(T') + q(T'))$
 - (c) $p(TT') - q(TT') = p(T) - q(T) + p(T') - q(T')$
 - (d) $p(e) = q(e) = 0$, and $p(S) = n$,

where S is any element of \mathcal{A} and n is the largest number not fixed under the permutation S . Note that (a) and (b) are equivalent because of (c).

(3) The commutation law holds. In a formal statement of this law, we should replace each of the elements of \mathcal{N} by a corresponding element of \mathcal{A} .

It is possible to show that to every relational system there corresponds an RSG defined in the obvious way with each matrix of the relational system corresponding to an element of the RSG. It is an open question whether to each RSG there corresponds a relational system.

Let \mathcal{J} and \mathcal{J}' be two RSG's. Then any mapping $h: \mathcal{J} \rightarrow \mathcal{J}'$ with the following three properties will be called an RSG homomorphism.

- (1) h preserves the product operation, i. e. for T_1 and T_2 in \mathcal{J} ,
 $h(T_1 T_2) = h(T_1)h(T_2)$.
- (2) h is the identity map from \mathcal{A} in \mathcal{J} to \mathcal{A} in \mathcal{J}' .
- (3) a. $p(T) \geq p(h(T))$,
 b. $q(T) \geq q(h(T))$,
 c. $p(T) - q(T) = p(h(T)) - q(h(T))$, for T in \mathcal{J} . Again,
 a and b are equivalent because of c.

If there is a homomorphism h from an RSG \mathcal{J} onto the RSG of a relational system \mathcal{R} , then we shall say that \mathcal{J} represents \mathcal{R} . Each T in \mathcal{J} will be said to represent its image $h(T)$ in \mathcal{R} .

The triple $(h, \mathcal{J}, \mathcal{R})$ may be regarded as analogous to a machine in ordinary automata theory. Here, an element T of \mathcal{J} corresponds to an input sequence. Its image $h(T)$ is the Boolean matrix describing the set of all pairs (i, j) such that T causes the transition from

state i to state j . There are also elements of \mathcal{J} whose images are a row vector and a column vector corresponding to the sets of initial and terminal states respectively.

It may be noted that an RSG plays the same role with respect to a relational system as does a theory to an algebra in the Eilenberg and Wright development [7].

A subset Γ of an RSG \mathcal{J} will be called recognizable if and only if \mathcal{J} represents a relational system \mathcal{R} with finite carrier K such that Γ represents just the true scalars in \mathcal{R} . We may show that this type of set is a generalization of a regular set in the automaton case. Similar closure properties may be proved to hold but the proofs are more difficult and not simple generalizations of the proofs in the automaton case.

Let \mathcal{R} be a relational system with carrier K and f any mapping from K onto some set K' . Then, for each matrix R in \mathcal{R} we may form a matrix R' on the carrier K' letting $[i'(1) \dots i'(p)r'j'(1) \dots j'(q)] = \Sigma[i(1) \dots i(p) r j(1) \dots j(q)]$, where the sum is taken over all matrix elements of R such that $f(i(1)) = i'(1), \dots, f(i(p)) = i'(p), f(j(1)) = j'(1), \dots, f(j(q)) = j'(q)$. If the image \mathcal{R}' of \mathcal{R} is a relational system and if the corresponding mapping is an RSG homomorphism, then we shall call f a relational system homomorphism from \mathcal{R} to \mathcal{R}' .

A somewhat different definition of homomorphism has been given by Yeh [45] for relational systems generated by $(1, 1)$ index matrices. His definition is more general in some respects and less general in others but fails to possess input-output symmetry.

Again, if we are given a relational system \mathcal{R} with carrier K , then we define a subset G of K to be recognizable if and only if there is a relational system homomorphism f from \mathcal{R} to a relational system \mathcal{R}' with finite carrier K' such that G consists of just those elements mapping to a given subset G' of K' .

In the case in which \mathcal{R} is an algebra the above definition may be shown to be equivalent to that given by Mezei and Wright [31]. Closure results continue to hold, however, in this generalized case. An important and interesting question concerns when the homomorphic image of a recognizable set is also recognizable. A partial answer to this question has been found.

2.7 Free Relational Semigroups

Let Ω be a set of letters w with a pair $(p(w), q(w))$ of nonnegative integers associated with each w in Ω . We may then form the free RSG generated by Ω which we write $\mathcal{J}(\Omega)$. It has the property that if h is any mapping from Ω to an RSG \mathcal{A} such that $p(w) \geq p(h(w))$, $q(w) \geq q(h(w))$, and $p(w) - q(w) = p(h(w)) - q(h(w))$ then h may be extended to an RSG homomorphism.

One may interpret each element T of $\mathcal{J}(\Omega)$ as a combinational network composed of blocks with labels w taken from Ω and having $p(w)$ inputs and $q(w)$ outputs. The network itself has $p(T)$ inputs and $q(T)$ outputs. Each input of a block is either an input of the network or else connected to an output of one other block. A similar rule holds for outputs. If $\mathcal{J}(\Omega)$ represents a relational system \mathcal{R}' , then for any T in

$\mathcal{H}(\Omega)$ we may regard the image $h(T)$ in as the operation carried out by the network T .

Another application of $\mathcal{H}(\Omega)$ is to language theory. Let Ω consist of a set \mathcal{A} of $(0, 1)$ symbols A , a single $(2, 1)$ symbol B , and a $(1, 0)$ symbol C . Then, we may use $\mathcal{H}(\Omega)$ to represent the free semigroup $\mathcal{L}(\mathcal{A})$ generated by \mathcal{A} . Each $A \in \mathcal{A}$ represents a row vector with a single unit corresponding to that generator in $\mathcal{L}(\mathcal{A})$, the element B represents the concatenation operation, and C represents a column vector consisting only of units. Since $\mathcal{L}(\mathcal{A})$ is an algebra, each row vector contains a single unit and represents the element or elements of $\mathcal{L}(\mathcal{A})$ corresponding to the position of that unit.

If Γ is recognizable on $\mathcal{H}(\Omega)$, then the row vectors R such that RC is represented by a member of Γ are just the strings in a context free language. Conversely, for every context free language on $\mathcal{L}(\mathcal{A})$ there is a recognizable Γ on $\mathcal{H}(\Omega)$. The finite relational system which is used to recognize Γ is just the context free grammar of the language. Productions in this grammar constitute a list of the unit (or true) matrix elements. The matrix element $[i(1)i(2)bj(1)] = 1$ of B is written as a production $j(1) \rightarrow i(1)i(2)$, and the matrix element $[aj(1)] = 1$ of $A \in \mathcal{A}$ is written $j(1) \rightarrow A$ where A is taken as a terminal symbol. Finally $[i(1)c] = 1$ of C is written $C \rightarrow i(1)$, where C is the initial symbol. This difference in convention is of little consequence. However, we may note that the elements of Γ may be regarded as trees, or bracketed strings while their images in $\mathcal{L}(\mathcal{A})$ have had

the brackets removed.

Another interesting avenue of study concerns relational systems in which the group \mathcal{A} is augmented by certain other matrices and the closure is taken under matrix multiplication. One such matrix is the column vector consisting entirely of units, already mentioned in connection with languages. Another is the bifurcator or $(\Gamma, 2)$ index matrix F in which $[i(1)fj(1)j(2)]$ is true if and only if $i(1) = j(1) = j(2)$. In the system of Eilenberg and Wright [7] both these operations are implicitly assumed. Finally, a third matrix is one which may be thought of as a union. This is a $(2, 1)$ index U in which $[i(1)i(2)uj(1)]$ is true if and only if either $i(1) = j(1)$ or $i(2) = j(1)$.

Section 3

Asymptotic Decomposability of Machines

3.1 Introduction

Series-parallel decomposition of sequential machines has been studied extensively from the point of view of the decomposition of specific machines [18, 23, 47]. Putzolu [36, 37] has recently obtained results concerning the likelihood that randomly chosen machines admit this decomposition. His analysis treated the case in which a state behavior type of realization was assumed and decomposition involving state splitting was not permitted. In the present paper we attack the analogous problem in which state splitting is allowed.

For definiteness and convenience we define here some of the concepts which will be needed in our study. Let $\llbracket n \rrbracket$ represent the set $\{1, \dots, n\}$, when n is any positive integer. Define a machine with n states and p inputs as any mapping δ from $\llbracket n \rrbracket \times \llbracket p \rrbracket$ to $\llbracket n \rrbracket$, and write $\llbracket n \rrbracket^{\llbracket n \rrbracket \times \llbracket p \rrbracket}$ to represent the set of all such mappings. Following [18], a decomposition of $\delta \in \llbracket n \rrbracket^{\llbracket n \rrbracket \times \llbracket p \rrbracket}$ into two machines $\delta_1 \in \llbracket n_1 \rrbracket^{\llbracket n_1 \rrbracket \times \llbracket p \rrbracket}$ and $\delta_2 \in \llbracket n_2 \rrbracket^{\llbracket n_2 \rrbracket \times \llbracket n_1 p \rrbracket}$ is specified by giving a mapping h from a subset of $\llbracket n_1 \rrbracket \times \llbracket n_2 \rrbracket$ onto $\llbracket n \rrbracket$ such that whenever $i = h(j, k)$ and $x \in \llbracket p \rrbracket$, then $h(\delta_1(j, x), \delta_2(k, (j-1)p + x))$ is defined and equals $\delta(i, x)$. A decomposition of δ into more than two machines is defined and equals δ if δ is defined and equals δ . A decomposition of δ into more than two machines is defined recursively as the result of further decomposing either δ_1 or δ_2 or both and repeating an arbitrary number of times. An

r -component decomposition is one which results from the performance of the basic decomposition process $r-1$ times and therefore yields r machines. We need not formulate here the well known associative principle which permits one to speak of the decomposition of δ into r machines $\delta_1, \dots, \delta_r$ without specifying the steps by which they were obtained. We shall say that δ is non-trivially decomposable if it admits an r -component decomposition in which each of the resulting r machines has fewer states than δ .

If one is interested only in state behavior realizations, then the mapping h is always taken as one-to-one. In this case to determine whether or not a machine is nontrivially decomposable one need only consider 2-component decompositions. However, in the more general case treated here there is no such restriction.

3.2 A Criterion for Decomposability

A machine $\delta \in \llbracket n \rrbracket^{\llbracket n \rrbracket \times \llbracket p \rrbracket}$ is called a group machine if each restriction $\delta(\cdot, x)$ is a permutation of $\llbracket n \rrbracket$. From any machine δ we may derive a group machine δ' by the simple expedient of letting $\delta'(\cdot, x) = \delta(\cdot, x)$ if $\delta(\cdot, x)$ happens to be a permutation and letting $\delta'(\cdot, x)$ be the identity map otherwise. We note that the corresponding group machine is the same as the original machine if and only if the original machine is a group machine.

Theorem 1

A machine δ is nontrivially decomposable if and only if the corresponding group machine δ' is nontrivially decomposable.

This theorem may be easily proved using results of Krohn and Rhodes [23] or of Zeiger [47]. Therefore we do not present a formal proof here. In following the construction of Zeiger one may decompose the machine δ into components δ_1 and δ_2 , where δ_2 has fewer states than δ , and δ_1 is a permutation reset machine of the same number of states as δ . The permutations of δ_1 correspond to those of δ and one may then show that to any nontrivial decomposition of δ there is an isomorphic nontrivial decomposition of δ_1 .

As a corollary to theorem 1 we have the result that if $\delta \in \llbracket n \rrbracket^{\llbracket n \rrbracket} \times \llbracket p \rrbracket$ is a machine of $n \geq 3$ states such that for no input $x \in \llbracket p \rrbracket$ is $\delta(\cdot, x)$ a permutation, then δ is nontrivially decomposable. It is this corollary which is required in the proof of theorem 2 in the next section. We note that in Zeiger's construction, if $\delta(\cdot, x)$ is never a permutation, then δ_1 is a reset machine. Then, any nontrivial partition on $\llbracket n \rrbracket = \llbracket n_1 \rrbracket$ has the substitution property for δ_1 . Hence δ_1 (and consequently δ) is nontrivially decomposable.

3.3 Decomposability of Randomly Selected Machines

In this section we shall be concerned with estimating what fraction of the set $\llbracket n \rrbracket^{\llbracket n \rrbracket} \times \llbracket p \rrbracket$ is nontrivially decomposable, and in particular shall investigate the way this function of n and p behaves as n and p approach ∞ .

Theorem 2

If $pn^{1/2}e^{-n} \rightarrow 0$ as $n \rightarrow \infty$, then the probability approaches 1 that a machine δ will be nontrivially decomposable if it is chosen at random from $[[n]]^{[[n]] \times [p]}$.

Proof: If $\delta \in [[n]]^{[[n]] \times [p]}$ is a machine such that for no $x \in [p]$ is $\delta(\cdot, x)$ a permutation then δ is surely decomposable by the following reasoning. Since $n \geq 2$ and $\delta'(\cdot, x)$ is the identity function for all x , we may decompose δ' in any way we wish using state behavior realizations and identity functions for the components. By theorem 1, δ is thus nontrivially decomposable. Now there are $n!$ permutations and n^n mappings of $[[n]]$ into $[[n]]$ so the probability that no mapping $\delta(\cdot, x)$ be a permutation is $(1 - \frac{n!}{n^n})^p$. However using Stirling's formula, we have $(1 - \frac{n!}{n^n})^p \geq 1 - \frac{pn!}{n^n} = 1 - \sqrt{2\pi} pn^{1/2}e^{-n} (1 + O(\frac{1}{n})) \rightarrow 1$ as $n \rightarrow \infty$. This completes the proof of the theorem.

Lemma 1

The fraction of all permutations of n letters in which no cycle has a length divisible by m is $\prod_{i=1}^{[n/m]} (1 - \frac{1}{im})$.

Proof: Write $T_m(n)$ for the number of permutations σ of $[[n]]$ having no cycle with a length divisible by m . For $r < m$, if the cycle containing the letter 1 has length r , then there are $T_m(n-r)$ ways in which the remaining cycles may be chosen. The cycle containing 1 may be chosen in $(n-1)!/(n-r)!$ ways. Hence, there are

$\sum_{r=1}^{m-1} [(n-1)!/(n-r)!] T_m(n-r)$ permutations of type σ in which the cycle containing 1 has length less than n . Now, the cycle containing 1 must not have length m , but if it has length greater than m , then we may treat the portion of this cycle consisting of 1 and the chain of m letters following 1 as if they were a single letter. This chain may be chosen in $(n-1)!/(n-m-1)!$ ways and following its choice there are $T_m(n-m)$ ways in which the remainder of the permutation may be chosen. Thus, we obtain

$$T_m(n) = \sum_{r=1}^{m-1} [(n-1)!/(n-r)!] T_m(n-r) + [(n-1)!/(n-m-1)!] T_m(n-m).$$

This difference equation has a unique solution subject to the conditions that $T_m(1) = 1$ and $T_m(0) = 1$, where it is assumed that coefficients for terms in $T_m(n-r)$ vanish when $r > n$. To prove the lemma, we show that

$$T_m(n) = n! \prod_{i=1}^{\lfloor \frac{n}{m} \rfloor} (1 - \frac{1}{im}) \text{ is a solution to the difference equation.}$$

We may check directly that $T_m(n) = n!$ when $n < m$ and also

$T_m(m) = (m-1)(m-1)!$. To justify our formula when $n > m$ it is more convenient to derive the following $(m+1)$ -st order difference equation:

$$nT_m(n-1) - T_m(n) = [(n-1)!/(n-m)!](n-m-1)[(n-m)T_m(n-m-1) - T_m(n-m)].$$

Substituting $T_m(n) = n! \prod_{i=1}^{\lfloor \frac{n}{m} \rfloor} (1 - \frac{1}{im})$, we may check that both sides

vanish when n is not divisible by m . Otherwise both sides have the

value $(n-1)! \prod_{i=j}^{\lfloor \frac{n-1}{m} \rfloor} (1 - \frac{1}{im})$. No spurious solution is possible since the first $m+1$ points agree with the solution to our original equation. This completes the proof of the lemma.

An immediate consequence of this lemma follows from the fact that when m is a prime or a power of a prime, the order of a permutation is divisible by m if and only if it has a cycle whose length is divisible by m . In this case, there are exactly

$T_m(n) = n! \prod_{i=1}^{\lfloor \frac{n}{m} \rfloor} (1 - \frac{1}{im})$ permutations of n letters whose order is not divisible by m . In our application the exact formula is less convenient than an asymptotic expression which may be derived as follows.

$$\begin{aligned} \prod_{i=1}^{\lfloor \frac{n}{m} \rfloor} (1 - \frac{1}{im}) &= \frac{\prod_{i=1}^{\lfloor \frac{n}{m} \rfloor} (i - \frac{1}{m})}{\lfloor \frac{n}{m} \rfloor!} \sim \frac{\lfloor \frac{n}{m} \rfloor^{-\frac{1}{m}}}{\Gamma(1 - \frac{1}{m})} \\ &\sim \frac{\lfloor \frac{n}{m} \rfloor^{-\frac{1}{m}}}{\Gamma(1 - \frac{1}{m})}. \end{aligned}$$

The asymptotic result holds when m remains fixed and $n \rightarrow \infty$, see reference [43]. Also, $n! \sim \sqrt{2\pi} n^{\frac{1}{2}} (\frac{n}{e})^n$, so we have

$$T_m(n) \sim \frac{m^{\frac{1}{m}} \sqrt{2\pi}}{\Gamma(1 - \frac{1}{m})} n^{(\frac{1}{m} - \frac{1}{m})} (\frac{n}{e})^n.$$

In particular, we obtain:

$$T_2(n) \sim 2 \left(\frac{n}{e}\right)^n, \text{ and } T_3(n) \sim 2.67n^{\frac{1}{6}} \left(\frac{n}{e}\right)^n.$$

The second of these expressions is used in the proof of theorem 3.

A group G of permutations on a set $[n]$ of n letters is called imprimitive [16] if there is a nontrivial partition on $[n]$ into disjoint subsets s_1, \dots, s_r with the substitution property [18]. This means that for all $g \in G$ and sets s_i in the partition, there is some set s_j in the partition such that $s_i g \subseteq s_j$. Since G is a group, one can easily show that actually $s_i g = s_j$. If there is no such partition, then G is called primitive and we have the following lemma.

Lemma 2

Two randomly chosen permutations on a set of n letters generate a primitive group with probability approaching 1 as n approaches ∞ .

Proof: Again write $[n] = \{1, \dots, n\}$ for the set of n letters to be permuted, and let σ and γ be chosen randomly from the set of $n!$ permutations of $[n]$. We see that the group generated by σ and γ is imprimitive if $[n]$ can be partitioned into two nonempty sets of cardinality n_1 , and $n - n_1$ such that σ and γ both permute the letters of these sets among themselves. Partitions of this type will be called separable, and we see that there are $(n_1!)^2 ((n - n_1)!)^2$ ways in which σ and γ may be chosen so that the partition into two such sets is separable. While it is clear that for a given σ and γ there may be more than one

separable partition, we have the upper bound

$$\sum_{n_1=1}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{n_1} (n_1!)^2 ((n - n_1)!)^2$$

number of ways σ and γ may be chosen so as to allow $\llbracket n \rrbracket$ to have a separable partition. Thus, of the $(n!)^2$ ways to choose σ and γ the fraction allowing a separable partition is no greater than

$$\sum_{n_1=1}^{\lfloor \frac{n}{2} \rfloor} \frac{n_1! (n - n_1)!}{n!} . \quad \text{The first two terms of this sum are } \frac{1}{n} + \frac{2}{n(n-1)}$$

while the $\lfloor \frac{n}{2} \rfloor - 2$ remaining terms are each no greater than $\frac{6}{n(n-1)(n-2)}$.

$$\text{We may therefore write } \sum_{n_1=1}^{\lfloor \frac{n}{2} \rfloor} \frac{n_1! (n - n_1)!}{n!} = \frac{1}{n} + O\left(\frac{1}{n^2}\right), \text{ and we ob-}$$

serve that this bound approaches 0 as n approaches ∞ .

Even if there is no separable partition of $\llbracket n \rrbracket$ there may yet be a partition with the substitution property. For this to occur, each class in such a partition must be capable of being mapped into any given class by a suitably chosen permutation in the group generated by σ and γ . Hence, all classes must contain the same number of letters. Let n_1 be the number of letters in a class and a the number of classes. Then $n = a n_1$ where $1 \leq a, n_1$ and there are $\frac{n!}{a! (n_1!)^a}$ ways of forming such a partition. For any given partition of this type the permutations

σ and γ may together be chosen in $[a!(n_1!)^a]^2$ ways so that the substitution property is satisfied. Thus $n! a! (n_1!)^a$ is an upper bound to the number of ways that σ and γ may be chosen so some partition into a equal classes has the substitution property. The ratio of this bound to $(n!)^2$ may be written

$$\frac{(n_1!)^a}{(a+1)(a+2) \dots n} = \left(\frac{2}{a+1}\right)\left(\frac{2}{a+2}\right) \dots \left(\frac{2}{2a}\right)\left(\frac{3}{2a+1}\right)\left(\frac{3}{2a+2}\right) \dots \left(\frac{3}{3a}\right) \dots$$

$$\left(\frac{n_1}{n-a+1}\right)\left(\frac{n_1}{n-a+2}\right) \dots \left(\frac{n_1}{n}\right).$$

We have taken $a > 1$ and as a consequence no factor in this product may be greater than $\frac{2}{3}$ and an upper bound to the product is therefore $\left(\frac{2}{3}\right)^{n-a} \leq \left(\frac{2}{3}\right)^{\left[\frac{n}{2}\right]}$. Since the number of possible choices for n_1 clearly cannot exceed n , there is the upper bound $n\left(\frac{2}{3}\right)^{\left[\frac{n}{2}\right]}$ to the probability that for randomly chosen permutations σ and γ there is a partition with the substitution property which is not separable. Since this bound also approaches 0 as n approaches ∞ the proof of the lemma is complete.

One may conclude slightly more than is contained in the statement of the lemma if a trivial refinement is made in the proof. The actual fraction of the pairs σ, γ having a separable partition with $n_1=1$ is less than $\frac{1}{n}$ but is asymptotic to $\frac{1}{n}$ as $n \rightarrow \infty$. Thus, we see that these partitions account for almost all cases in which the group generated by σ and γ is imprimitive as $n \rightarrow \infty$. The probability that

this group will be imprimitive is thus asymptotic to $\frac{1}{n}$.

Theorem 3

If $pn^{\frac{1}{\delta}}e^{-n} \rightarrow \infty$ as $n \rightarrow \infty$, then the probability approaches 0 that a machine δ will be nontrivially decomposable if it is chosen at random from $[[n]]^{[[n]]} \times [[p]]$.

Proof: By the theory of Krohn and Rhodes [23], if there exist inputs $x \in [[p]]$ such that the corresponding mappings $\delta(\cdot, x)$ are permutations which generate the alternating group A_n on $n \neq 4$ letters, then the machine δ is indecomposable. This fact follows because some component in any decomposition of δ must have an input semigroup with a subgroup which is the inverse homomorphic image of A_n . Therefore, this component must have at least n states, so δ fails to satisfy our criterion of nontrivial decomposability. We shall show that the probability approaches 1 as n approaches ∞ that A_n is a subgroup of the input semigroup of δ .

Let s be any permutation of $[[n]]$ which cyclically permutes some set $\{i, j, k\} \subseteq [[n]]$ of three states and produces a permutation of some order ν on the remaining states which is not divisible by 3. Then s may be used to generate a cyclic permutation t of $\{i, j, k\}$, since we may take $t = s^\nu$ which cyclically permutes $\{i, j, k\}$, leaving the remaining letters fixed. We compute that there are $\frac{n(n-1)(n-2)}{3}$ cyclic permutations of some set $\{i, j, k\} \subseteq [[n]]$. Furthermore, for each such permutation there are $T_3(n-3)$ permutations of the remaining states whose order ν is not divisible by 3. Hence, there are $\frac{n(n-1)(n-2)}{3} T_3(n-3)$ permutations

of type s. By lemma 1 and the asymptotic expression for $T_3(n)$ derived from it, we have

$$T_3(n-3) \sim K(n-3)^{\frac{1}{6}} \left(\frac{n-3}{e}\right)^{n-3} \sim K n^{\frac{1}{6}-3} \left(\frac{n}{e}\right)^n, \text{ where } K = \frac{3^{\frac{1}{3}} \sqrt{2\pi}}{\Gamma\left(\frac{2}{3}\right)} = 2.67.$$

Thus, $\frac{n(n-1)(n-2)}{3} T_3(n-3) \sim \frac{K}{3} n^{\frac{1}{6}} \left(\frac{n}{e}\right)^n$. The probability that none of the first $\left[\frac{p}{3}\right]$ inputs yields a permutation of type s is therefore

$$\begin{aligned} \left(1 - \frac{n(n-1)(n-2)T_3(n-3)\left[\frac{p}{3}\right]}{3 n^n}\right) &< e^{-\frac{\left[\frac{p}{3}\right] n(n-1)(n-2)T_3(n-3)}{3 n^n}} \\ &\sim e^{-\frac{K}{9} p n^{\frac{1}{6}} e^{-n}} \rightarrow 0, \text{ as } n \rightarrow \infty. \end{aligned}$$

Hence, the probability approaches 1 that at least one of the inputs $x \in \left[\left[\frac{p}{3}\right]\right]$ yields a permutation $s = \delta(\cdot, x)$ which may be used to generate a cyclic permutation t of three states.

The probability also approaches 1 that each of the two sets $\left[\left[\frac{2p}{3}\right]\right] = \left[\left[\frac{p}{3}\right]\right]$ and $\left[\left[p\right]\right] = \left[\left[\frac{2p}{3}\right]\right]$ contains some input yielding a permutation because these sets contain on the order of $\frac{p}{3}$ inputs. By lemma 2, we note that two randomly chosen permutations of $\left[\left[n\right]\right]$ generate a primitive group G with probability approaching 1 as n approaches ∞ . Hence, we assume that a primitive group G on $\left[\left[n\right]\right]$ is generated from two of the inputs in $\left[\left[p\right]\right] = \left[\left[\frac{2p}{3}\right]\right]$ and that a cyclic permutation t of three states is generated from one of the inputs in $\left[\left[\frac{p}{3}\right]\right]$.

Let \mathcal{U} be the set of all elements of the group $G \cup \{t\}$ which cyclically permute three states, leaving all others fixed. It is known [16] that if \mathcal{U} contains all such cyclic permutations, then A_n is a subgroup of $G \cup \{t\}$. Define $\theta \subseteq [n] \times [n]$ as the set of all pairs (i_1, i_2) both lying in some cycle of an element of \mathcal{U} . Then θ is an equivalence relation since if (i_1, i_2) lie on a cycle of t_1 and (i_2, i_3) lie on a cycle of t_2 , then (i_1, i_3) lie on a cycle $(i_1 i_2 i_3) = t_1 t_2^{-1} t_1^{-1} t_2$ if the third members of the cycles for (i_1, i_2) and (i_2, i_3) are distinct and on a cycle $(i_1 i_2 i_3) = t_2 t_1$ if the third members are the same. This construction shows us that whenever i_1, i_2 , and i_3 are in the same θ class, then the cycle $(i_1 i_2 i_3)$ is in \mathcal{U} . Also, θ induces a partition on $[n]$ which has the substitution property with respect to G . For, let g be any element of G , then $(i_1, i_2) \in \theta$ with corresponding $t_1 \in \mathcal{U}$, we see that $g^{-1} t_1 g$ is also a cyclic permutation in g for the pair $(i_1 g, i_2 g)$. Consequently $(i_1 g, i_2 g)$ is in θ . But G was assumed to be primitive and \mathcal{U} is nonempty since it contains t , so \mathcal{U} must contain all cyclic permutations of three states. Thus, \mathcal{U} generates A_n , with probability approaching 1 as n approaches ∞ . This completes the proof.

3.4 Additional Remarks

The condition for nontrivial decomposability stated in section 3.1 applies to each input of a machine separately. Therefore, for fixed n , the probability of nontrivial decomposability can never increase as p increases. However, the theorems of section 3.3 fail to provide information concerning the probability of nontrivial decomposability in

certain cases as indicated by the following corollary to theorems 2 and 3.

Corollary 1

If $\frac{n - \ln p}{\ln n}$ approaches a limit ℓ as n approaches ∞ then the probability of nontrivial decomposability approaches 1 if $\ell > \frac{1}{2}$ and approaches 0 if $\ell < \frac{1}{6}$.

Proof: If $\lim_{n \rightarrow \infty} \left(\frac{n - \ln p}{\ln n} \right) > \frac{1}{2}$, then for some $\epsilon > 0$ and n_0 we have $\frac{n - \ln p}{\ln n} > \frac{1}{2} + \epsilon$ whenever $n \geq n_0$. Hence, $p n^{\frac{1}{2} + \epsilon} e^{-n} < 1/n^\epsilon$ and since $1/n^\epsilon \rightarrow 0$ as $n \rightarrow \infty$, we may apply theorem 2. If $\lim_{n \rightarrow \infty} \left(\frac{n - \ln p}{\ln n} \right) < \frac{1}{6}$, then for some $\epsilon > 0$ and n_0 we have $\frac{n - \ln p}{\ln n} < \frac{1}{6} - \epsilon$ whenever $n \geq n_0$. Hence, $p n^{\frac{1}{6} - \epsilon} e^{-n} > n^\epsilon$ and since $n^\epsilon \rightarrow \infty$ as $n \rightarrow \infty$, we may apply theorem 3. This completes the proof.

We note that nothing is known about the case in which ℓ lies in the range $\frac{1}{6} < \ell < \frac{1}{2}$. It is conjectured that this "gap" is removable and that a more delicate pair of theorems similar to 2 and 3 is valid in which the same exponents of n appear. In particular, it seems possible that the condition $p n^{\frac{1}{2}} e^{-n} \rightarrow \infty$ may be sufficient to obtain the conclusion of theorem 3.

Corollary 2

If $\frac{n}{\ln p}$ approaches a limit f' as n approaches ∞ , then the probability of nontrivial decomposability approaches 1 if $f' > 1$ and approaches 0 if $f' < 1$.

This result follows easily from corollary 1, and permits one to compare the probability of this type of decomposition with state behavior decomposition. In [36, 37] it is shown that the limiting value of $\frac{\ln n}{n}$ is an appropriate quantity to consider in order to determine whether the limiting probability of nontrivial state behavior decomposition is 1 or 0.

Another question of some interest concerns whether or not a machine δ is nontrivially decomposable into just two components δ_1 and δ_2 , each having fewer states than δ . As was pointed out in the introduction, a machine may be nontrivially decomposable without being nontrivially decomposable into two components. An example is the universal 4-state machine which is decomposable into 4 components of 3, 2, 2 and 3 states, but not into two components each of fewer than 4 states.

According to [10] a machine admits a nontrivial 2-component decomposition if and only if it admits a nontrivial set system decomposition.

Using methods similar to those used in the proof of theorem 2, it is possible to show that if $\frac{n - \ln p}{\ln n} \geq \frac{3}{2}$ as $n \rightarrow \infty$, then for any $k > 0$, the probability that a randomly chosen $\delta \in \mathbb{N}^{\mathbb{N}} \times \mathbb{P}^{\mathbb{P}}$ admits at least k nontrivial SP set systems approaches 1; correspondingly, the expected number of nontrivial SP set systems of a machine approaches ∞ .

An unsettled question is whether set system (i. e. 2-component) nontrivial decomposability is almost surely satisfied if

$\frac{3}{2} \leq \frac{n - \ln p}{\ln n} < \frac{1}{2}$ as $n \rightarrow \infty$. It is hoped that further studies may provide answers to some of these questions.

Section 4

The Recognizability of Equational Sets

Automata theory and the theory of formal languages are closely related: automata recognize, parse and translate languages; languages are used to program and describe automata. This research is concerned with some specific results relating the generation and recognizability of languages.

It is well known to automata theorists that a language is generated by a left linear grammar iff it is recognized by some finite automaton. In this case, there is an effective process whereby one can find a deterministic system (i. e. finite automaton) which recognizes the language generated by a non-deterministic system (i. e. left linear grammar). It has been shown by Mezei and Weight [31] that this effective process can be generalized to any case where the languages in question are subsets of the set T of all fully parenthesized legal algebraic expressions or computation trees on some fixed set of operator symbols Ω (each symbol in Ω has a given constant specifying the number of operands required). In this generalization, "generated by a left linear grammar" is replaced by "generated by a context free grammar which allows only legal algebraic expressions (on Ω and some finite set of nonterminals) to be substituted for nonterminals." (Such sets are called equational by Mezei and Wright [31], grammatical by Muller [33], and algebraic by Eilenberg and Wright [20]. They might also be called algebraic context free languages.).

"it is recognized by some finite automaton" is replaced by "it is the inverse homomorphic image in T of a subset of a finite algebra on Ω ." (Such sets are called "recognizable" by Mezei and Wright [31], and "finite tree automaton definable" by Rabin [38]. Notice that T must be made into an algebra on Ω for homomorphism to be defined; this can be done in a natural way, and the resulting algebra is called the generic algebra on Ω .

The definitions of equationality and recognizability can, in fact, be made in any algebra \mathcal{A} on Ω . Equational sets in \mathcal{A} are precisely the homomorphic images of equational sets in T . Recognizable sets in \mathcal{A} are the inverse homomorphic images in \mathcal{A} of subsets of finite algebras on Ω . It is now of interest to ask for which algebras the equational sets and the recognizable sets coincide.

In any finitely generated algebra, recognizable sets are equational. However, in a finitely generated free semigroup the equational sets are the usual context free languages while the recognizable sets are just the regular events. Hence there exist finitely generated algebras in which not all equational sets are recognizable. Thus the following conditions on a finitely generated algebra \mathcal{A} on Ω , each of which by itself implies that the equational sets of \mathcal{A} are recognizable, are of interest:

- 1) \mathcal{A} is isomorphic to a generic algebra modulo a congruence ρ_G defined by an algebraic phrase structure grammar G by $(x, y) \in \rho_G$ iff $x \xrightarrow{*}_G y$ and $y \xrightarrow{*}_G x$. (See

Ginsburg [15] for definitions of phrase structure grammar and $\xrightarrow{*}_G$; when the strings being operated are legal algebraic expressions on Ω and some set of nonterminals, G is called algebraic iff the rules of G allow only replacement of one legal algebraic subexpression by another).

This condition includes the case where \mathcal{A} is finitely presentable on a generic algebra. The following is used in the proof of sufficiency: Theorem: Every algebraic phrase structure language in a generic algebra is equational.

- 2) \mathcal{A} is isomorphic to a generic monadic algebra modulo a congruence which is a binary transduction. (See Elgot and Mezei [8] for the definition of binary transduction.) This includes the monadic case of (1) but is not included by it, as may be seen using a monadic algebra with idempotent operators. The latter is of interest in asynchronous switching theory. The condition above has been generalized to non-monadic algebras.
- 3) \mathcal{A} is a subset of an algebra \mathcal{B} on some set Ω' of operator symbols, where the equational subsets of \mathcal{B} are recognizable, and where certain relations hold between the operations defined on \mathcal{A} by Ω and the operations defined on \mathcal{B} by Ω' . This result is due to Muller [33].

If the equational sets of an algebra \mathcal{A} are recognizable, then all word problems on \mathcal{A} are solvable. The converse, unfortunately,

does not hold.

It is thought that these results have some implications for parenthesis languages, loop free parallel computation, data structures (lists in particular) and possibly other areas.

Section 5

Algebraic Isomorphism Invariants for Transition Graphs

The material presented here summarizes and concludes research already appearing in the 1966 and 1967 annual reports. In the meantime these results were issued as Systems Engineering Laboratory Technical Report No. 21, "Algebraic Isomorphism Invariants for Transition Graphs," where full proofs and bibliographical references are to be found. Here an effort has been made to present in a self-contained and compact form the more significant results.

A transition graph G is a finite, directed graph such that every point of G has outdegree 0 or 1. (By a directed graph $G = (X, \gamma)$ we mean the graph of a relation γ on a set X . Thus "loops" are permitted.) Thus $G = (X, \gamma)$ is a transition graph if and only if γ is a partial transformation on X . Accordingly, we say that a transition graph $G = (X, \gamma)$ is a transformation graph if γ is a (complete) transformation on X and that $G = (X, \gamma)$ is a permutation graph if γ is a permutation on X .

The composition $G \circ G'$ of two graphs $G = (X, \gamma)$ and $G' = (X, \gamma')$ (G and G' are defined on the same set of points) is the result of composing their lines, i. e.

$$G \circ G' = (X, \gamma\gamma')$$

where $\gamma\gamma'$ is the composition of the relations γ and γ' .

Let \mathcal{K}_n denote the set of all digraphs on the points $N_n = \{1, 2, \dots, n\}$. Then obviously (\mathcal{K}_n, \circ) is a semigroup. If further we let $\mathcal{M}_n(F)$ denote the multiplicative semigroup of linear

transformations on an n -dimensional vector space V over a field F , the natural representation of \mathcal{H}_n relative to F and some representation basis $\mathcal{A} = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ (\mathcal{A} is any basis for V) is the function:

$$\rho: \mathcal{H}_n \rightarrow \mathcal{T}_n(F)$$

where $\rho(G) = T_G$ is defined as follows:

$$\alpha_i T_G = \sum_{j=1}^n a_{ij} \alpha_j \text{ where } a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in \gamma \\ 0 & \text{otherwise,} \end{cases}$$

for all $\alpha_i \in \mathcal{A}$. (Thus, the natural representation of \mathcal{H}_n relative to F and \mathcal{A} is simply a linear transformation equivalent of the usual representation of graphs by adjacency matrices. More precisely, if the adjacency matrix A_G of G is regarded as being over the representation field F then A_G is just the matrix of T_G with respect to the representation basis \mathcal{A} .)

In terms of the natural representation we observe, first of all, the following well known result (usually stated for adjacency matrices over the reals):

Theorem 1

If two graphs G and G' are isomorphic ($G \cong G'$) then, under the natural representation (relative to any choice of representation field F and basis \mathcal{A}), the representing transformations T_G and $T_{G'}$ are similar ($T_G \sim T_{G'}$).

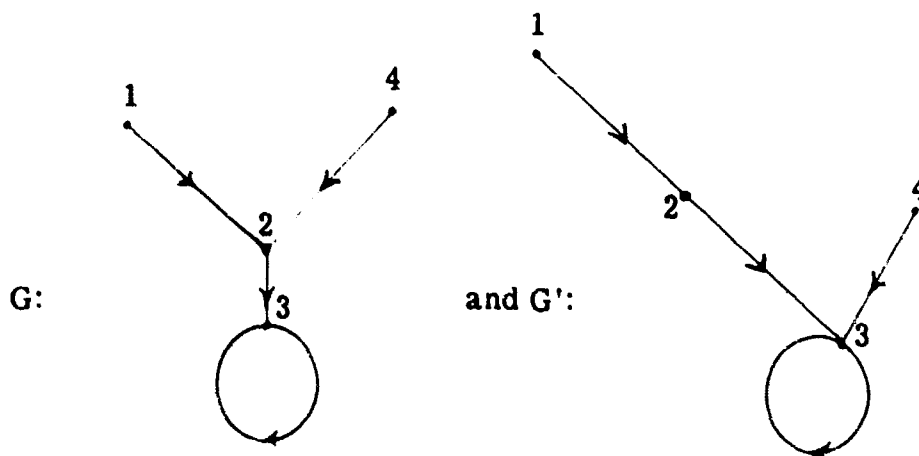
Note that the condition " $T_G \sim T_{G'}$ " is stronger than that of being "cospectral" since the latter requires only that T_G and $T_{G'}$ have the same characteristic polynomial. In particular, for example, one can show that all n -point directed trees which are either to or from a point must be cospectral and yet many such trees can be distinguished by the fact that their representations are nonsimilar.

In what follows we assume that the natural representation is restricted to the semigroup \mathcal{U}_n of transition graphs on N_n .

Theorem 2

There exist nonisomorphic transition graphs that are similarly represented.

Proof: Consider the following two graphs in \mathcal{U}_4 :



and suppose that the representation field is the reals. Then one can show that T_G and $T_{G'}$ have the same nontrivial invariant polynomials (invariant factors), namely

$$x^3 - x^2 \text{ and } x$$

and so $T_G \sim T_{G'}$. As G and G' are obviously nonisomorphic, the theorem holds.

Thus, even for this relatively restricted class of graphs, a complete set of similarity invariants for the representing linear transformations (e. g. their invariant polynomials or elementary divisors) fails to yield a complete set of isomorphism invariants. The investigation summarized below is concerned with the discovery of just why this is so, the main result being a graphical characterization of the structural information conveyed by any complete set of similarity invariants.

Summarizing, first of all, some well known results regarding the general structure of transition graphs, every (weak) component of a transition graph is either a flower (a weakly connected transformation graph) or a tree-to-a-point. (Trees-to-a-point will subsequently be referred to as "trees.") Also, transition graphs are obviously unipathic and consequently we can use the notation $[x, y]$ to denote a path from x to y . $l[x, y]$ will denote the length of path $[x, y]$. A cycle-point of a transition graph is any point that lies in a cycle. A tree-point is any point that is not a cycle-point. Thus every point of a transition graph G is a cycle-point iff G is a permutation graph; every point of G is a tree-point iff G is a forest. The period of a flower G is the number of

cycle-points of G (i. e. the length of its unique cycle). The root of a tree G is the unique point of G having outdegree 0.

If G is a transition graph and x is a point of G , let $C(x)$ denote the (weak) component determined by x . Then the notion of "height," as usually defined for trees, can be extended to transition graphs as follows:

Definition 1

If $G = (X, \gamma)$ is a transition graph and $x \in X$ then the height $h(x)$ of x is defined as follows:

i) If $C(x)$ is a flower then

$$h(x) = \min \{l[x, y] \mid y \text{ is a cycle-point of } C(x)\}$$

ii) If $C(x)$ is a tree then

$$h(x) = l[x, x_0] \text{ where } x_0 \text{ is the root of } C(x).$$

(Note that $h(x) = 0$ if and only if x is either a cycle-point or a root.)

Definition 2

The height $h(G)$ of a transition graph G is the maximum height of any point of G .

For connected¹ transition graphs (i. e. trees and flowers), we find that the invariant polynomials of the representing transformations are intimately related to the heights of certain points in the corresponding -----

¹"Connected" will subsequently mean "weakly connected" unless otherwise qualified.

graphs. This important relationship can be expressed in the form of an algorithm for computing the invariant polynomials of T_G directly from the structure of G . If $G = (X, \gamma)$ is a transition graph let $R(G, x)$ denote the reachable set of x , i. e.

$$R(G, x) = \{y \mid [x, y] \text{ is a path of } G\}$$

and if Y is a proper subset of X let $G - Y$ denote the removal of Y from G , i. e.

$$G - Y = G \text{ restricted to the set of points } X - Y.$$

Then given any transition graph G we define a sequence of subgraphs

$$G_1, G_2, \dots, G_t$$

as follows:

- i) $G_1 = G$
- ii) If x_i is a point of maximum height in $G_i = (X_i, \gamma_i)$ and $R(G_i, x_i) \neq X_i$ then

$$G_{i+1} = G_i - R(G_i, x_i).$$

Otherwise the sequence terminates, that is, $G_t = G_i$. We say that such a sequence is derived from G and although a derived sequence is not necessarily unique (even up to isomorphism) we obtain the following important result.

Theorem 3

If G is a connected transition graph and G_1, G_2, \dots, G_t is a sequence of subgraphs derived from G then the representing linear transformation T_G has t nontrivial invariant polynomials $\psi_i(x)$, $i=1, 2, \dots, t$, which can be graphically determined as follows:

i) If G is a flower of period r then

$$\psi_1(x) = x^{h(G_1)+r} - x^{h(G_1)}.$$

If G is a tree then

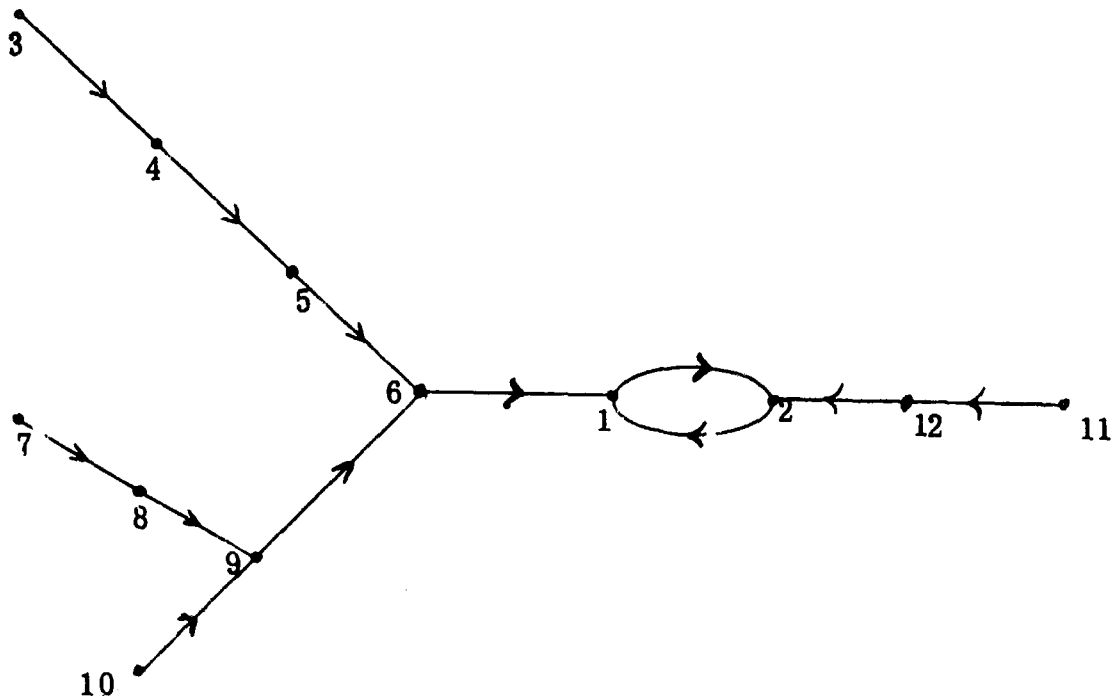
$$\psi_1(x) = x^{h(G_1)+1}.$$

ii) If $t > 1$ then

$$\psi_i(x) = x^{h(G_i)+1}, \quad i=2, 3, \dots, t.$$

The proof of the theorem is based on the classical decomposition of a vector space V (relative to a linear transformation T on V) into cyclic subspaces V_1, V_2, \dots, V_t such that the minimum polynomial of V_i coincides with the i -th nontrivial invariant polynomial of T . The process of forming a derived sequence of subgraphs parallels this decomposition process where points of maximum height correspond to vectors which generate the various cyclic subspaces. The graph obtained on removing a maximum reachable set $R(G_1, x_1)$ corresponds to the linear transformation \bar{T} induced by T on the quotient space $V/V_1 \oplus V_2 \oplus \dots \oplus V_i$. Therefore, although a somewhat lengthy proof is required to take care of all the details, the verification is conceptually rather straightforward.

To illustrate the theorem, consider the following transition graph which is a flower of period 2 on 12 points:

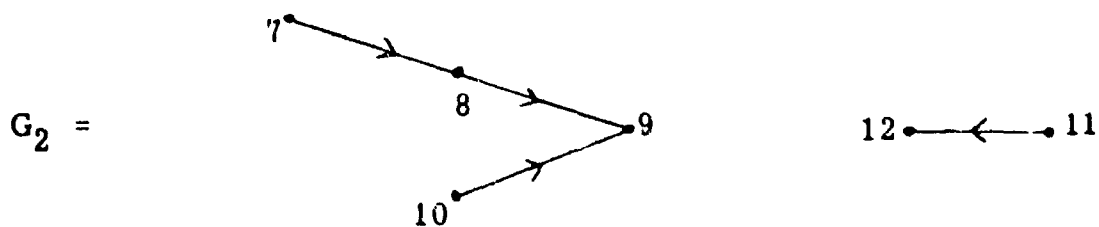


Forming a derived sequence of subgraphs:

$$G_1 = G$$

and as $h(3) = h(G_1)$ (we could have also chosen point 7) and

$R(G_1, 3) = \{1, 2, 3, 4, 5, 6\}$ we have:



As $h(7) = h(G_2)$:

$$G_3 = \begin{array}{c} \bullet \\ 10 \end{array} \quad \begin{array}{c} \bullet \quad \longleftarrow \quad \bullet \\ 12 \qquad \qquad 11 \end{array}$$

Removing $R(G_3, 11)$,

$$G_4 = \begin{array}{c} \bullet \\ 10 \end{array}$$

and as $R(G_4, 10) = \{10\}$, the process terminates. Accordingly, T_G has four nontrivial invariant polynomials, namely

$$\psi_1(x) = x^6 - x^4$$

$$\psi_2(x) = x^3$$

$$\psi_3(x) = x^2$$

and $\psi_4(x) = x.$

If we remove the connectedness constraint, the procedure of the previous theorem cannot, in general, be applied. However, if we require that the transition graph be a forest¹ then such a generalization is possible, that is:

¹In what follows we will use the term "forest" to mean "transition graph forest." Thus G is a forest iff every component of G is a tree-to-a-point.

Theorem 4

If G is a forest and G_1, G_2, \dots, G_t is a sequence of subgraphs derived from G then T_G has t nontrivial invariant polynomials $\psi_i(x)$ where

$$\psi_i(x) = x^{h(G_i)+1}, \quad i=1, 2, \dots, t.$$

In order to obtain a graphical characterization of the invariant polynomials for arbitrary transition graphs, we first define two sequences of numerical invariants that relate directly to the structure of a transition graph G . If x is a tree-point of G the depth $d(x)$ of x is the length of the longest (directed) path to x , i. e.

$$d(x) = \max\{l[y, x] \mid [y, x] \text{ is a path of } G\}.$$

Accordingly

Definition 3

If $G \in \mathcal{U}_n$ then the depth sequence of G is the sequence

$$\delta(G) = (d_0, d_1, \dots, d_{n-1})$$

where

$$d_j = \text{the number of tree-points } x \text{ of } G \text{ such that } d(x) = j,$$

$$j=0, 1, \dots, n-1.$$

Thus if $\delta(G) = (d_0, d_1, \dots, d_{n-1})$, the sum

$$t = \sum_{j=0}^{n-1} d_j$$

is simply the total number of tree-points of G . In particular,

if G is a forest then $t=n$; if G is a permutation graph,
 $t=0$.

A second sequence describes the cycle structure of a transition graph and is defined as follows:

Definition 4

If $G \in \mathcal{U}_n$ the period sequence of G is the sequence

$$\pi(G) = (r_1, r_2, \dots, r_n)$$

where

r_j = the number of components of G that are
 flowers of period j ,

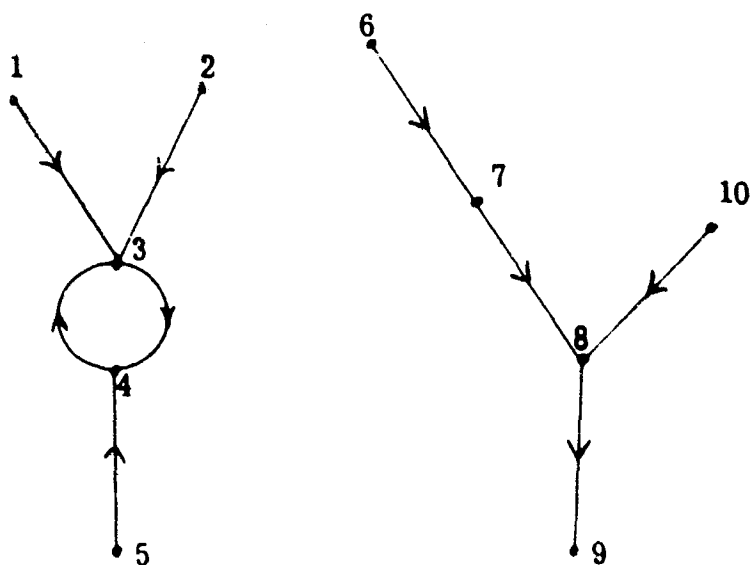
$j=1, 2, \dots, n$.

Note that when G is a permutation graph, $\pi(G)$ corresponds
 to the usual description of cycle structure for permutations.

At the other extreme, if G is a forest then $\pi(G) = (0, 0, \dots, 0)$.

Example: The transition graph

G :



has depth sequence

$$\delta(G) = (5, 1, 1, 1, 0, 0, 0, 0, 0, 0)$$

and period sequence

$$\pi(G) = (0, 1, 0, 0, 0, 0, 0, 0, 0, 0).$$

The study of transition graphs decomposes rather naturally into the study of two important subclasses, namely forests and permutation graphs. If we suppose first that G is a forest, it follows that T_G is nilpotent and, accordingly, for some integer $1 \leq t \leq n$, T_G has t invariant polynomials

$$x^{e_1}, x^{e_2}, \dots, x^{e_t}$$

where $e_i \geq e_{i+1}$ ($1 \leq i < t$). (The integers e_i are sometimes referred to as the indices of a nilpotent linear transformation.) By applying theorem 4, the indices of a representing nilpotent transformation T_G can be related directly to the structure of G as follows:

Theorem 5

If G is a forest and T_G has indices e_1, e_2, \dots, e_t

then the depth sequence of G is

$$\delta(G) = (d_0, d_1, \dots, d_{n-1})$$

where $d_j = |\{i \mid e_i > j\}|$, $j=0, 1, \dots, n-1$.

In other words the number of points of G having depth

j is equal to the number of invariant polynomials of T_G

having degree greater than j .

Turning this result around we can show, conversely, that the depth sequence uniquely determines the invariant polynomials of T_G

that is,

Theorem 6

If G is a forest with depth sequence

$$\delta(G) = (d_0, d_1, \dots, d_{n-1})$$

then, for each integer i ($1 \leq i \leq n$), i occurs exactly

$$m_i = d_{i-1} - d_i$$

times as an index of T_G .

Combining the previous two results we have proved that forests are similarly represented if and only if they have the same depth sequence, i. e.

Theorem 7

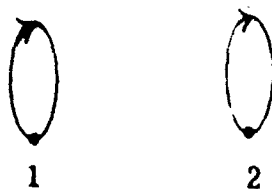
If G and G' are forests then $T_G \sim T_{G'}$ if and only if

$$\delta(G) = \delta(G').$$

This, then, completes the graphical characterization of similarity for forests.

To this point, the transition graphs considered have been such that similarity invariants of their corresponding linear transformations do not depend on the nature of the representation field. However, when we consider transition graphs having nontrivial period sequences (i. e., multi-flower transition graphs) this is no longer the case. To illustrate this fact, consider the following two-component permutation graph:

G :



If the representation field F is a field of characteristic 0 (say, the reals) then the characteristic polynomial ϕ_{T_G} is given by:

$$\phi_{T_G}(x) = x^2 - 2x + 1.$$

On the other hand, if $F = F_2$, the 2-element field, then

$$\phi_{T_G}(x) = x^2 + 1.$$

Consequently, if G is compared with the connected permutation graph



which, over any field, is represented by a transformation having the characteristic polynomial

$$\phi_{T_{G'}}(x) = x^2 - 1,$$

if F has characteristic 0 then we see that the nonisomorphic graphs G and G' can be distinguished by their corresponding characteristic polynomials. On the other hand, if the representation field is F_2 , we observe that T_G and $T_{G'}$ have the same characteristic polynomial (i. e. G and G' are cospectral over F_2). An important question, therefore, is whether the graphical interpretation of a complete set of similarity invariants (the characteristic polynomial is generally incomplete) will likewise depend on the choice of representation field. We begin the investigation of this question by graphically formulating the elementary divisors of T_G relative to the various possible choices of

a representation field F .

In determining the elementary divisors of T_G we note first of all that we can restrict our attention to prime fields (a field is prime if it contains no proper subfields). This is possible since T_G is defined in terms of the scalars 0 and 1 and, consequently, if F is a representation field of characteristic k ($k=0$ or some prime p) T_G is also over the prime subfield F_k of F . As two linear transformations are similar over F_k if and only if they are similar over any extension of F_k (i. e., any field of characteristic k), no loss of generality will result from such a restriction.

Let F_k denote a prime field of characteristic k and let $\Phi_i(x)$ denote the i -th cyclotomic polynomial (over F_k) where i is any positive integer not divisible by k ($k \nmid i$). (If $k=0$, $\Phi_i(x)$ is defined for all $i \geq 1$. $\Phi_i(x)$, by definition, is the polynomial whose roots are all the primitive i -th roots of unity found in any extension of F_k .) If we suppose now that G is a permutation graph and consider first the case where $F = F_0 = \mathbb{Q}$ (the rational numbers) the elementary divisors of T_G can be graphically determined as follows:

Theorem 8

If G is a permutation graph with period sequence

$$\pi(G) = (r_1, r_2, \dots, r_n)$$

and the natural representation is over \mathbb{Q} then, for all

i such that $1 \leq i \leq n$, the cyclotomic polynomial $\Phi_i(k)$

occurs exactly

$$m_i = \sum_{i|j} r_j$$

times as an elementary divisor of T_G . Moreover, when taken in their totality, these are all the elementary divisors of T_G .

To express a similar result for fields of prime characteristic p , if j is some positive integer, let $\epsilon(j)$ denote the exponent of p the prime decomposition of j (if $p \nmid j$, $\epsilon(j) = 0$). Thus j can be written:

$$j = j'p^{\epsilon(j)}$$

where $p \nmid j'$. If further we let $\delta_{i,j}$ denote the familiar Kronecker delta then

Theorem 9

If G is a permutation graph with period sequence

$$\pi(G) = (r_1, r_2, \dots, r_n)$$

and the natural representation is over F_p (p a prime)

then, for all i such that $1 \leq i \leq n$ and $p \nmid i$, each primary¹ factor of the polynomial

$$\phi_i(x)^{p^k}$$

occurs exactly

$$m_i^{(k)} = \sum_{i|j} \delta_{k, \epsilon(j)} r_j$$

times as an elementary divisor of T_G ($k=0, 1, \dots, [\log_p p]$).

¹In the prime decomposition of a polynomial, the primary factors are the various powers of the prime factors.

Moreover, when taken in their totality, these are all the elementary divisors of T_G .

For each of the formulations given by the theorems 8 and 9, if one now supposes that the integers m_i or $m_i^{(k)}$ are given and regards the formulae as a system of equations in the n unknowns

$$r_1, r_2, \dots, r_n$$

it can be shown, in each case, that the system has a unique solution.

(The proof for the characteristic 0 case is immediate whereas the proof for the characteristic p case is somewhat more involved.) This proves

Theorem 10

If G is a permutation graph then over an arbitrarily chosen prime field (and hence over any field) the elementary divisors of T_G uniquely determine the period sequence of G .

Combining the previous three results, it follows that two permutation graphs are similarly represented if and only if they have the same period sequence, i. e.,

Theorem 11

If G and G' are permutation graphs then $T_G \sim T_{G'}$ if and only if $\pi(G) = \pi(G')$.

Having graphically characterized similar representations for forests in theorem 7, and permutation graphs in theorem 11, we now obtain a general solution by showing that an arbitrary transition graph can always be analyzed in terms of a suitably determined forest and/or

permutation graph. Let $G = (X, \gamma)$ be a transition graph and let X_F denote the set of all tree-points of G and X_P the set of all cycle-points of G . Then

Definition 5

The reduction \bar{G} of a transition graph $G = (X, \gamma)$ is the graph

$$\bar{G} = (X, \bar{\gamma})$$

where

$$\bar{\gamma} = \gamma - \{(x, y) \mid (x, y) \in \gamma, x \in X_F, y \in X_P\}.$$

In short, \bar{G} is the result of removing all the lines of G that are from a tree-point and to a cycle-point. Alternatively, if we let G_F denote the restriction of G to X_F (assuming $X_F \neq \emptyset$) and let G_P denote the restriction of G to X_P (assuming $X_P \neq \emptyset$) then G_F is a forest, G_P is a permutation graph and G can be described as follows:

$$\bar{G} = \begin{cases} G_F = G & \text{if } G \text{ is a forest} \\ G_P = G & \text{if } G \text{ is a permutation graph} \\ G_F + G_P & \text{otherwise.} \end{cases}$$

Thus any transition graph that contains both tree and cycle points reduces to the (direct) sum of a forest and a permutation graph. The justification of this reduction is complete once we establish the following important property.

Theorem 12

If G is a transition graph and \bar{G} is the reduction of G then, over an arbitrarily chosen representation field,

$$T_G \sim T_{\bar{G}}.$$

Given the above property along with the fact that

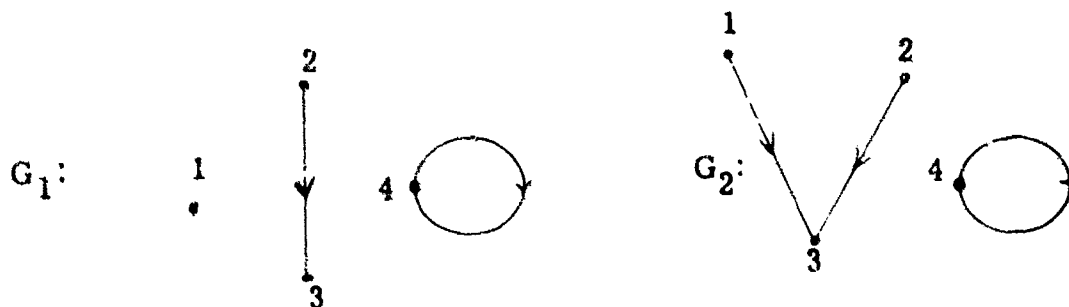
- i) $\bar{G} = G_F + G_P$ (assuming G is neither a forest nor a permutation graph)
 - ii) The depth sequences $\delta(G)$ and $\delta(G_P)$ are essentially the same;
and
 - iii) The period sequences $\pi(G)$ and $\pi(G_P)$ are essentially the same,
- we are able to combine the characterizations obtained for forests and permutation graphs and establish the main result of our investigation.

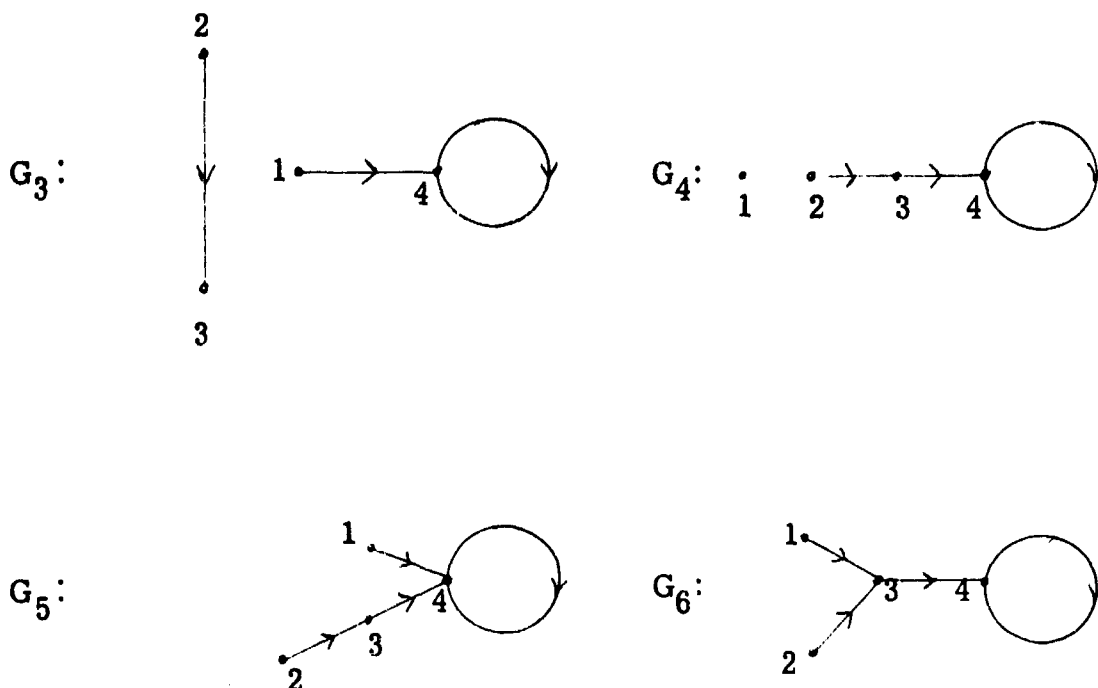
Theorem 13

If G and G' are transition graphs then $T_G \sim T_{G'}$ if and only if $\delta(G) = \delta(G')$ and $\pi(G) = \pi(G')$.

In other words, two transition graphs with n points are similarly represented (over an arbitrary field F) if and only if they agree both in the number points that are tree-points of depth j ($0 \leq j \leq n-1$) and in the number of components that are flowers of period k ($1 \leq k \leq n$).

Example: Each of the following nonisomorphic transition graphs has depth sequence $(2, 1, 0, 0)$ and period sequence $(1, 0, 0, 0)$ and consequently the representing transformations of all six graphs are similar.





Applying the previous theorem, it becomes relatively easy to distinguish subclasses of transition graphs for which a complete set of similarity invariants yields a complete set of isomorphism invariants. To be more precise, let

$$\mathcal{I}_n = \{f_1, f_2, \dots, f_n\}$$

denote the set of isomorphism invariants defined on \mathcal{U}_n as follows:

$$f_i(G) = \psi_i(x), \text{ the } i\text{-th invariant polynomial of } T_G,$$

$$i=1, 2, \dots, n.$$

We refer to \mathcal{I}_n as the set of rational isomorphism invariants and, as a consequence of the previous theorem, obtain the following criterion for completeness:

Theorem 14

If $\mathcal{H} \subseteq \mathcal{U}_n$ then \mathcal{H}_n is a complete set of isomorphism invariants for \mathcal{H} if and only if

$$\delta(G) = \delta(G') \text{ and } \pi(G) = \pi(G') \implies G \cong G'$$

for all G, G' in \mathcal{H} .

Applying this criterion, it is immediate that:

Theorem 15

The rational isomorphism invariants are complete for the class

$$\mathcal{Z}_n = \{G \mid G \in \mathcal{U}_n, h(G) = 0\}$$

of all zero-height transition graphs.

As \mathcal{Z}_n includes class \mathcal{A}_n of permutation graphs on N_n we have

Corollary 15.1

If $G, G' \in \mathcal{A}_n$ then $G \cong G'$ if and only if $T_G \sim T_{G'}$
(over an arbitrary representation field).

(The corollary is known for representations over a field of characteristic 0. On the other hand, the generalization to fields of prime characteristic appears to be new.)

Completeness classes containing graphs with nontrivial tree structures have also been discovered, the most interesting being a generalization of "homogeneous" trees. We say that a transition graph is homogeneous if points of the same height have the same indegree and the same outdegree. Every homogeneous transition graph is either a forest or a transformation graph and by a rather straightforward

argument one can prove:

Theorem 16

The rational isomorphism invariants are complete for
the class \mathcal{H}_n of homogeneous transition graphs.

Stating this result in terms of the representing linear transformations
we have:

Corollary 16.1

If $G, G' \in \mathcal{H}_n$ then $G \cong G'$ if and only if

$$T_G \sim T_{G'}.$$

Section 6

Iterative Network Realization of Sequential Machines

This research is a continuation of the study of a class of iterative networks which was described in the 1967 annual report. It is primarily concerned with the realization of sequential machines using a special type of cellular network. This section will summarize the results obtained during the past year. Proofs and additional details can be found in SFL technical report no. 06920-22-T.

In the 1967 report, some of the cellular networks that have appeared in the literature were examined and classified according to interconnection structure and cell type. The special class of "iterative networks" was then informally described in terms of these classifications. This class of networks was formalized by means of the following mathematical model.

Definition 1

An (abstract) iterative network is a 6-tuple

$N = (G, X, \eta, S, \Sigma, \delta)$ where:

1. G is a finite group called the interconnection group of N with $n = |G|^{\dagger}$.
2. X is a subset of G with $k = |X|$.
3. η is a one-to-one function, called the group ordering for G , mapping I_n onto G such that the image of I_k is $X^{\dagger\dagger}$.

 $^{\dagger} |G|$ denotes the cardinality of the set G .

$^{\dagger\dagger} I_n$ denotes the set $\{1, 2, \dots, n\}$.

4. S is a finite set of internal symbols with $|S| \geq 2$.
5. Σ is a finite set of inputs symbols with $|\Sigma| \geq 1$.
6. δ is a function mapping $S^k \times \Sigma$ into S and called the cell function of N .

The following definition allows an iterative network to be interpreted as a sequential machine.

Definition 2

Let $N = (G, X, \eta, S, \Sigma, \delta)$ be an iterative network. Then the machine realized by N is the sequential machine

$M(N) = (S^G, \Sigma, \bar{\delta})$ where:

1. S^G , the set of all functions from G to S , is the set of all states.
2. Σ is the set of input symbols.
3. $\bar{\delta}$ is the state transition function mapping $S^G \times \Sigma$ into S^G , and defined as follows. For f in S^G and σ in Σ , the successor state of f under input σ is the function $(f, \sigma)\bar{\delta}$ which takes an element g of G into the element

$$[(gx_1)f, \dots, (gx_k)f, \sigma]\bar{\delta}$$

of S where $i\eta = x_i$ for $i=1, 2, \dots, k$.

The function $\bar{\delta}$ is called the behavior of the network N .

There are several notations in the literature of one machine imitating another. When the machines in question do not have outputs, then the only thing which must be imitated is the state transition function. This is usually done by finding states or sets of states in the imitating machine which behave in the same way, with respect to their respective transition functions, as the states of the machine being imitated. We will use the following notion of one machine imitating another.

Definition 3

Let M and M' be sequential machines. Then M is said to realize M' if there is a submachine of M isomorphic to M' .

This definition is frequently weakened even more by only requiring that M' be a homomorphic image of a submachine of M . However, for our use, the stronger version is sufficient.

It is by means of this concept of imitation that the abstract theory of machines is usually linked to the physical circuits used to realize them. In our case, the circuit has been modeled as an iterative network which realizes sequential machines as follows.

Definition 4

Let M be a sequential machine, N an iterative network, and $M(N)$ the machine realized by N . Then the network N realizes the machine M if and only if $M(N)$ realizes M .

Note that any iterative network N will always realize $M(N)$, the machine realized by N . In general, it will also realize many other machines as well. Hence the machine realized by the iterative network N is the sequential machine $M(N)$ (definition 3) while a machine realized by N is any machine isomorphic to a submachine of $M(N)$.

In order to find an iterative network which realizes a given sequential machine, it is first necessary to find a group which can be used as the network's interconnection group. The following result suggests that we should use some subgroup of the machine's automorphism group.

Theorem 1

The interconnection group G of the iterative network N is isomorphic to a subgroup of the automorphism group of $M(N)$.

The next result states that, in fact, any subgroup may be used.

Theorem 2

Let $M = (Q, \Sigma, \lambda)$ be a sequential machine and G be any group of automorphisms of M . Then there is an iterative network N which uses G as its interconnection group and which realizes M .

There are two cases in which Theorem 2 results in "trivial" realizations. The first is when the automorphism group of the machine to be realized is trivial. This results in a network with only one cell which is as complex as the machine. The second case arises when the cardinality of the set of internal symbols is equal to the number of states in Q . In this case, all of the cells are as complex as the original machine and nothing is gained by using more than one cell.

It has been shown that the second case can always be avoided if the number of states in Q is greater than 2. However, both cases can be avoided simultaneously by extending Theorem 2 to obtain a "binary network".

An iterative network is called binary if the cardinality of its set of internal symbols is two. That is, if $N = (G, X, \eta, S, \Sigma, \delta)$ is an iterative network, then N is a binary network if and only if $|S| = 2$. In this case, every cell has a single output terminal which can take on only two distinct values. We ask the following question. Given an arbitrary sequential machine, can we find a binary iterative network which realizes it? The answer is given by the following theorem.

Theorem 3

Given any sequential machine $M = (Q, \Sigma, \lambda)$, there exists at least one binary iterative network which realizes it.

The proof of this result involves first embedding the machine to be realized in a larger machine which has a nontrivial automorphism group. Then the techniques used in the proof of Theorem 2 are used to realize this machine. Since the relation of machine realization is transitive, this resulting network will realize the original machine.

Although the proof of Theorem 3 is constructive, it can not be considered a practical design technique due to the large number of cells in the realizing network. This procedure can, however, be applied successfully to certain special classes of machines to get more practical binary realizations. In particular, the class of autonomous machines and the class of total automata (i. e. strongly connected sequential machines whose automorphism group is as large as possible) can be realized with binary networks where the number of cells is the same order of magnitude as the number of states in the machine. In addition, total automata can be realized with binary networks where the number of internal cell input terminals is equal to the number of external input symbols.

Section 7

Optimal Sequencing of Jobs Subject to Deadlines

7.1 Introduction

Held and Karp [19] have developed a general dynamic programming algorithm for problems involving the sequencing of jobs subject to deadlines. There are a number of these problems, however, for which a certain "consistency principle" holds. This principle can be stated as follows:

Given a set of jobs which are to be processed subject to deadlines, there exists a linear ordering of the complete set of jobs, such that for any subset of these jobs, an optimal sequence exists in which the ordering of the jobs completed on time is consistent with the linear ordering.

This paper demonstrates how this consistency principle can be exploited to obtain solution methods which are considerably more efficient than that of Held and Karp. Problems considered are (i) a single machine and jobs with individual deadlines

(Moore [32]), (2) the same problem with partial ordering restrictions on the sequence, (3) two machines in series and jobs with a common deadline (Johnson [20]), and (4) a single machine, jobs with a common deadline, and individual linear deferral costs (McNaughton [30] and Smith [41]). These problems are dealt with in Sections 2, 3, 4, and 5, respectively. Two concluding sections discuss additional problems with partial ordering restrictions and multi-machine generalizations.

7.2 Single Machine. Multiple Deadlines

Let there be given jobs J_1, J_2, \dots, J_n . For each job J_i , let t_i denote its processing time on a single machine, D_i its deadline, and r_i a reward which is earned if processing of the job is completed by the deadline. Without loss of generality, assume that $D_i \leq D_{i+1}$; for $i=1, 2, \dots, n-1$.

We wish to find a sequence for the jobs which maximizes the sum of the earned rewards. According to Moore [32], there exists an optimal sequence in which (1) the jobs completed on time (meeting deadlines) precede the tardy jobs and (2) the on-time jobs are ordered according to their deadlines, earliest deadline first.

It follows that the problem consists of making a selection of the jobs which are to be completed on time. Given such a selection, the ordering of these jobs is determined solely by their deadlines;

the ordering of the remaining (tardy) jobs which follow is arbitrary. The selection problem can be formulated as an integer linear program as follows:

$$\text{Maximize } \sum_{i=1}^n r_i x_i$$

$$\text{subject to } Tx \leq D$$

where

$$\begin{aligned} x_i &= 1 \text{ if job } J_i \text{ is selected} \\ &= 0 \text{ otherwise,} \end{aligned}$$

and T and D are of form

$$T = \begin{bmatrix} t_1 & 0 & 0 & & 0 & 0 & 0 \\ t_1 & t_2 & 0 & \dots & 0 & 0 & 0 \\ t_1 & t_2 & t_3 & & 0 & 0 & 0 \\ \vdots & & & & \vdots & & \\ t_1 & t_2 & t_3 & & t_{n-2} & 0 & 0 \\ t_1 & t_2 & t_3 & \dots & t_{n-2} & t_{n-1} & 0 \\ t_1 & t_2 & t_3 & & t_{n-2} & t_{n-1} & t_n \end{bmatrix}, \quad D = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{n-2} \\ D_{n-1} \\ D_n \end{bmatrix}$$

As one might expect from the special structure of this problem, it can be solved by dynamic programming techniques similar to those used for the well known "knapsack" problem.

Definition: Let $f_i(t)$ = the maximum attainable total reward for a selection of jobs from the set of jobs $\{J_1 \dots J_i\}$, subject to the constraint that the completion time of no job is later than t .

A recursion relation for $f_i(t)$ can be formulated as follows:

- 1) Consider $t \leq D_i$. If there exists an optimal schedule in which J_i is completed on time, then J_i can be the last on time job and $f_i(t) = f_{i-1}(t-t_i) + r_i$. If no such schedule exists, then $f_i(t) = f_{i-1}(t)$.
- 2) For $t \geq D_i \geq \dots \geq D_1$, $f_i(t) = f_i(D_i)$ and can be computed in the manner indicated above.

Hence, for $i=1, 2, \dots, n$:

$$f_i(t) = f_i(D_i) \quad \text{for } t > D_i$$

$$f_i(t) = \max \{f_{i-1}(t), r_i + f_{i-1}(t-t_i)\} \quad \text{for } 0 < t \leq D_i$$

subject to the boundary conditions

$$f_i(0) = 0$$

$$f_0(t) = 0$$

$$f_i(t) = -\infty \quad \text{for } t < 0$$

The maximum attainable total reward for the complete set of jobs is given by $f_n(D_n)$.

Assuming that all of the processing times and deadlines are integers, the length of the computation grows no more rapidly than

nD_n , i. e. proportional to the product of the number of jobs and the longest deadline.

Note also that the problem is symmetric with respect to time. This means that with slight modification of the recursion equations, one can just as well solve a problem in which the times D_i denote earliest possible starting times for the jobs, and one determines a sequence which maximizes the sum of the earned rewards, subject to a common deadline, t , for all jobs.

7.3 Single Machine Problems with Partial Ordering Restrictions

In the single machine problem just discussed, it was assumed that any sequence for the jobs was feasible. An additional complication arises, however, when it is required that the sequence for the jobs must be consistent with a given partial ordering Q , which may be due to technological restrictions of various kinds.

First we shall develop necessary and sufficient conditions for the existence of a sequence in which all jobs are completed on time, consistent with the given partial ordering restrictions. These conditions generalize the result of Smith [41] that an unrestricted set of jobs can all be completed on time if and only if they are completed on time when they are sequenced according to deadlines.

Again let there be given jobs J_1, J_2, \dots, J_n , where job J_i has processing time t_i and deadline D_i , with an arbitrary partial ordering Q . The object is to find a linear ordering L

such that all jobs can be processed on time consistent with Q
iff they are on time in the sequence defined by L .

Assume, without loss of generality, that $D_1 \leq D_2 \leq \dots \leq D_n$
and for each job, J_i , define a vector, V^i , as follows:

$$V^i = (V_1^i, V_2^i, \dots, V_n^i)$$

where $V_k^i = 1$ if $(J_i, J_k) \in Q$
 $= 0$ otherwise.

Let L denote the relation on the jobs obtained by ordering
these vectors lexicographically from the largest to the smallest.

Lemma 1: L is a linear ordering.

Proof: Since L is determined by a lexicographic ordering, it is
known to be both reflexive and transitive. Hence, it is only neces-
sary to show that L is antisymmetric. Consider a pair of jobs,
 J_i and J_k , $i \neq k$ where $V^i \geq V^k$. We must show that $V^i > V^k$.

Suppose $V^i = V^k$, then $V_i^k = V_k^i = 1$, since Q is a partial ordering.

Hence (J_i, J_k) and $(J_k, J_i) \in Q$, a contradiction since Q is a
partial ordering and $i \neq k$.

Hence $V_i \neq V_k$ and $V_i > V_k$.

Lemma 2: L is consistent with Q .

Proof: If $(J_i, J_k) \in Q$, then $V_k^i = 1$ implies that $V_i^i = 1$.

Further, $V_i^k = 0$ while $V_i^i = 1$. Hence $V^i > V^k$ and $(J_i, J_k) \in L$.

Theorem: All jobs in the set J can be completed on time consistent with the partial ordering Q iff they are on time in the sequence defined by L .

Proof: If - Obvious

Only if:

Consider an arbitrary sequence $S = (J_{i_1} \dots J_{i_n})$ in which all jobs are completed on time consistent with the partial ordering Q , where S is distinct from L . Then there exists a pair of adjacent jobs, $J_{i_k} J_{i_{k+1}}$ in the schedule S where

$$V^{i_{k+1}} > V^{i_k}.$$

Thus, there must be a first position, say q , in which the vector $V^{i_{k+1}}$ contains a 1, but the vector V^{i_k} contains a 0. Clearly, $q \neq i_k$ and there are two cases to consider:

$$1) \quad q = i_{k+1}$$

Then $D_{i_{k+1}} \leq D_{i_k}$ and a new schedule, S_1 may be defined from S by interchanging J_{i_k} and $J_{i_{k+1}}$ in which J_{i_k} and $J_{i_{k+1}}$ are still on time.

$$2) \quad q < i_{k+1}$$

Then job J_q has $D_q \leq D_{i_{k+1}}$, D_{i_k} and $(J_{i_{k+1}}, J_q) \in Q$.

Hence J_q follows both $J_{i_{k+1}}$ and J_{i_k} in the schedule S and a new schedule, S_1 may be defined from S by interchanging J_{i_k} and $J_{i_{k+1}}$ in which J_{i_k} and $J_{i_{k+1}}$ are still on time.

If S_1 is L, we are done. Otherwise the above process is repeated to obtain schedules $S_2 \dots S_p$ where S_p is L.

Q. E. D.

This result can be used to obtain a dynamic programming solution to the following problem. All jobs are to be completed on time, subject to the partial ordering restriction Q, where for each job J_i there is a choice of one of m processing times $t_i^1, t_i^2, \dots, t_i^m$ with associated rewards $r_i^1, r_i^2, \dots, r_i^m$. (It is reasonable to assume that the rewards would be proportional to the processing times.)

Since all jobs are to be completed on time consistent with Q, the only sequence that need be considered is that defined by L. Consequently, let us now assume that the jobs are reindexed according to L, and let $f_i(t)$ denote the maximum attainable total reward associated with the first i jobs in the sequence L, subject to the condition that no job is completed later than time t.

A recursion equation for $f_i(t)$ is as follows. For $i=1, 2, \dots, n$:

$$\begin{aligned} f_i(t) &= f_i(D_i) \quad \text{for } t > D_i \\ &= \max_{j=1, 2, \dots, m} \{r_i^j + f_{i-1}(t-t_i^j)\} \quad \text{for } 0 \leq t \leq D_i, \end{aligned}$$

subject to the boundary conditions

$$f_i(0) = 0$$

$$f_0(t) = 0$$

$$f_i(t) = -\infty \quad \text{for } t < 0.$$

The case in which it is not required that all jobs are completed on time is discussed in Section 6.

7.4 Two Machines in Series, Common Deadline

We now deal with the problem of two machines in series and a common deadline where each job must be processed by both machines, in sequence. Consider a set of jobs, $J = \{J_1, \dots, J_n\}$ where for each job J_i , a_i denotes its processing time on the first of two machines, and b_i its processing time on the second. Let r_i denote a reward which is earned if job J_i is completed by the second machine by a deadline T (common to all jobs). The problem is to find a sequence for the set of jobs which maximizes the sum of the earned rewards.

It follows from the results of the previous chapter and those of Johnson [20] that there exists an optimal sequence in which (1) the jobs completed on time precede the tardy jobs; (2) the jobs are processed in the same order by both machines; and (3) the on-time jobs are ordered according to the following relation:

Job J_p precedes J_q only if $\min \{a_p, b_q\} \leq \min \{a_q, b_p\}$.

Once again, the problem consists of making a selection of the jobs which are to be completed on time. Given such a selection,

the ordering of these jobs is determined by Johnson's relation;
the ordering of the remaining jobs which follow is arbitrary.

Without loss of generality, assume that

$$\min \{a_j, b_{j+1}\} \leq \min \{a_{j+1}, b_j\}, \text{ for } j=1, \dots, n-1.$$

Let $f_i(t_1, t_2)$ = the maximum attainable reward for a selection of jobs from among J_1, \dots, J_i , subject to the constraint that the completion time of no job is later than time t_1 on the first machine or t_2 on the second. Following the type of argument used in Section 2, a recursion relation for $f_i(t_1, t_2)$ can be formulated as follows.

For $i=0, 1, 2, \dots, n$:

$$f_i(t_1, t_2) = \max \{f_{i-1}(t_1, t_2), r_i + f_{i-1}(\min\{t_1 - a_i, t_2 - a_i - b_i\}, t_2 - b_i)\}$$

subject to the boundary conditions:

$$f_0(t_1, t_2) = 0$$

$$f_i(0, 0) = 0$$

$$f_i(t_1, t_2) = -\infty \quad (t_1 < 0 \text{ or } t_2 < 0)$$

The maximum attainable total reward for the complete set of jobs is, of course, given by $f_n(T, T)$. Assuming all processing times are integers, the length of the computation grows no more rapidly than nT^2 .

7.5 Single Machine, Common Deadline, Linear Deferral Costs

As a further example of the consistency principle, consider the problem of a single machine, a common deadline, and linear deferral costs.

Consider a set of jobs $J = \{J_1, \dots, J_n\}$, where for each job J_i , a_i denotes its processing time and r_i a reward which is earned only if the job is completed by a deadline T (common to all jobs). In addition, let p_i denote a linear deferral cost coefficient. If the job J_i is completed at a time $t \leq T$, the net reward earned for that job (reward minus deferral cost) is $r_i - p_i t$.

Consider the position of a contractor who is free to accept or reject jobs. For each job J_i which is accepted and completed prior to the deadline, a reward r_i is earned. However, the deferral of the job causes a cost to be incurred which is determined by the coefficient p_i . What selection of jobs maximizes profit?

Given any selection of jobs, such that the sum of their processing times is no greater than T , the jobs should be ordered according to the ratios p_i/a_i , the job with the largest ratio being processed first. This result has been found by McNaughton [30] and Smith [41].

Without loss of generality, assume $\frac{p_i}{a_i} \leq \frac{p_{i+1}}{a_{i+1}}$. Let $f_i(t)$ = the maximum attainable total net profit for a selection of jobs from among J_1, J_2, \dots, J_i , subject to the constraint that the starting time of the first job is t and the last job is completed no

later than T .

An appropriate set of recursion equations is as follows for $t \geq 0$.

$$f_i(t) = \max \{f_{i-1}(t)r_i - p_i(t+a_i) + f_{i-1}(t+a_i)\}$$

subject to the boundary conditions

$$f_i(T) = 0$$

$$f_0(t) = 0$$

$$f_i(t) = -\infty \quad \text{for } t > T.$$

The maximum attainable total net profit for the complete set of jobs is given by $f_n(0)$. Assuming all processing times are integers, the length of the computation grows as nT .

There is an interesting variation of this problem in which the deadline is not controlling. E.g., T is as large as the sum of all the processing times. In this case, the selection of jobs is controlled solely by the question of whether or not jobs can be completed before their deferral costs exceed their rewards.

7.6 Additional Partial Ordering Considerations

It would be desirable to be able to extend the results of Sections 2, 4, and 5 to the situation in which optimal sequences are to be derived subject to the restriction that either (a) all jobs are ordered consistently with an arbitrary partial ordering Q or (b) all on time jobs are so ordered. (Section 3 presented such results for the case in which all jobs were required to be on time.)

Unfortunately, the results obtained above do not generalize except for very special partial orderings for which the consistency principle can be shown to hold.

One such case arises when the partial ordering induces a partitioning of the jobs into a sequence of equivalence classes, (E_1, \dots, E_p) where

- 1) For each equivalence class, E_i , $J_r, J_s \in E_i$,
implies that J_r and J_s are not related by Q .
- 2) $J_r \in E_i, J_s \in E_{i+1}$ implies that $(J_r, J_s) \in Q$.

In this case, the dynamic programming algorithms previously defined may be applied to each equivalence class individually and the optimal sequence can be obtained by concatenating the resulting sequences for E_1, E_2, \dots, E_p . (The individual solutions, of course, take into account that the starting time for the first job in the i -th equivalence class equals the sum of the processing times for the jobs in the previous $i-1$ equivalence classes.)

There are also other situations in which Q may not be of this special form, but the parameters of the problem are such that the consistency principle can be shown to hold.

7.7 Machines in Parallel

Each of the problem formulations and solution methods given above can be extended in a very natural way to the situation in which there are many machines, or sets of machines, in parallel.

and any given job can be processed by any given machine. In each such extension, the jobs that are assigned to any given machine are processed in an order which is consistent with the ordering obtained by solving the associated single machine problem.

Consider the extension of the problem of multiple deadlines (Section 2 above). Let there be given a set of jobs, $J = \{J_1 \dots J_n\}$. For each job J_i , let $a_{i,k}$ denote its processing time on the k -th of M machines and $r_{i,k}$ a reward which is earned if processing of the job is performed on machine k and completed prior to the deadline for the job, D_i .

As before, we assume without loss of generality, that $D_i \leq D_{i+1}$, for $i=1, 2, \dots, n-1$. (Note that it is feasible to have different deadlines on different machines. However, it must be the case that $D_i \leq D_{i+1,k}$ for all k .) Let $f_i(t_1 \dots t_n)$ = the maximum attainable reward for a selection of jobs from among J_1, J_2, \dots, J_i , subject to the constraint that the completion time of no job is later than t_k for machine k . Now, we have, for $i=0, 1, \dots, n$:

$$\begin{aligned} f_i(t_1 \dots t_m) &= f_i(t_1, t_2, \dots, t_{k-1}, D_i, t_{k+1}, \dots, t_m), \text{ if } t_k > D_i \\ &= \max \{f_{i-1}(t_1, t_2, \dots, t_m), \\ &\quad \max_k \{r_{i,k} + f_{i-1}(t_1, t_2, \dots, t_{k-a_{i,k}}, \dots, t_m)\}\} \\ &\quad \text{otherwise} \end{aligned}$$

subject to the boundary conditions

$$f_0(t_1, t_2, \dots, t_m) = 0$$

$$f_i(0, 0, \dots, 0) = 0$$

$$f_i(t_1, t_2, \dots, t_m) = -\infty \quad \text{if any } t_k < 0$$

The length of the computation implied by these recursion equations grows as mnD_n^m . Some saving, of course, can be realized through exploitation of symmetry in the case in which all machines are identical, i. e. $a_{j,p} = a_{j,q}$ and $r_{j,p} = r_{j,q}$ for all j, p, q .

The formulation of recursion equations for the extensions of the problem described in Sections 4 and 5 is quite similar, and results in computations which grow as mnT^{2m} and mnT^m , respectively. One should compare the extension of the deferral cost case with the solution method given by Rothkopf [40] for the multi-machine deferral cost problem without deadlines. In that case, a computational growth of mnT^{m-1} is possible.

We note that in the generalizations of the problems of Sections 4 and 5, it is possible to have some variation in the characteristics of the individual machines, provided the existence of a single linear ordering is not interfered with. In the case of sets of two machines in series, it must be possible to find a linear ordering such that, for all k ,

$$\min(a_{j,k}, b_{j+1,k}) \leq \min(a_{j+1,k}, b_{j,k})$$

and in the case of linear deferral costs,

$$\frac{p_{j,k}}{a_{j,k}} \leq \frac{p_{j+1,k}}{a_{j+1,k}}$$

The rewards, $r_{j,k}$, need be related in no particular way.

Section 8

THE THEORY OF FORMAL LANGUAGES AND ITS IMPACT ON THE DESIGN AND IMPLEMENTATION OF PROGRAMMING LANGUAGES

Leonard Y. Liu

8.1 Introduction

Recently, research in the area of formal languages has been intensified because of interesting interpretations with respect to programming languages. A formal language is any set of finite length strings of symbols over a finite alphabet. The theory of formal languages is concerned with the description of languages, their properties, structures, relationships and their recognition. Some of the results of this theory are of direct interest to the designer of computers, programming languages and compilers. Some of these results are briefly reviewed in this section.

8.2 Finite Descriptions of Languages

Formal languages were first considered by mathematical linguists as mathematical models for natural languages [5]. If a natural language had only a finite number of sentences, it could be completely specified by a finite list. However, this is not the case. Almost every meaningful language has an infinite number of sentences.

Chomsky defined four types of grammars [3, 4].

A type 0 grammar is a 4-tuple $G = (N, \Sigma, P, S)$. N and Σ are two disjoint nonempty sets of nonterminal symbols and terminal

symbols respectively. S is a special nonterminal symbol (sentence symbol). P is a finite nonempty set of production rules of the form $\alpha \rightarrow \beta$. α is any string of nonterminal symbols and terminal symbols with at least one nonterminal symbol in it. β is any string of terminal and nonterminal symbols. Let $\omega_1, \omega_2, \alpha, \beta, \gamma_1, \gamma_2$ be strings of terminal symbols and nonterminal symbols. If $\omega_1 = \gamma_1 \alpha \gamma_2$, $\omega_2 = \gamma_1 \beta \gamma_2$, and $\alpha \rightarrow \beta$ is in P , then $\omega_1 \Rightarrow \omega_2$ is used to denote this fact ($\omega_1 \Rightarrow \omega_2$ is used whenever G is understood). $\omega_0 \xRightarrow{*} \omega_n$ if there exist $\omega_1, \dots, \omega_n$ such that $\omega_i \xRightarrow{*} \omega_{i+1}$ for $1 \leq i < n$ ($\omega_0 \xRightarrow{*} \omega_n$ is used whenever G is understood). The language generated by G is denoted by $L(G)$.

$$L(G) = \{x \mid S \xRightarrow{*} x \text{ and } x \text{ is a string of terminal symbols}\}.$$

A type 1 grammar (context sensitive grammar), G , is a restricted type 0 grammar in that the length of β is longer than or equal to the length of α for every production rule $\alpha \rightarrow \beta$ in G . A language is a context-sensitive language if it can be generated by a context sensitive grammar.

A type 2 grammar (context free grammar), G , is a restricted type 1 grammar in that every production rule in G is of the form $A \rightarrow \beta$, where A is a single nonterminal symbol and β is a string of terminal and nonterminal symbols. A language is called a context-free language if it can be generated by a context free grammar.

A type 3 grammar, G , is a restricted type 2 grammar in that every production rule is of the form $A \rightarrow aB$ or $A \rightarrow a$ where A, B are

nonterminal symbols and a is a terminal symbol.

It is well known that the context free grammars form the backbone of the syntax of programming languages. The most representative example of this is the use of context free grammar in specifying most of the syntactic rules of ALGOL 60 [34]. Clearly, this permits a more precise and efficient specification of the programming languages. However, attention should be paid to the fact that context free grammars are not quite powerful enough to completely specify most programming languages. Parts which are not appropriate for context free specifications are specified by English in ALGOL 60 [34]. On the other hand, the context sensitive grammars can completely specify the programming languages. An interesting problem is to define some class of languages, lying between the class of context sensitive languages and the class of context free languages, that is powerful enough to model the programming languages and may be parsed efficiently. Little is known about classes of languages which lie between the class of context sensitive languages and the class of context free languages. Indexed grammars [1], programmed grammars [39], and finite-reversal pushdown automata [27] specify classes of such languages.

8.3 Ambiguity

When a programming language is being designed, it is always essential to know whether the "interpretation" of every sentence in the language is unique or not. This idea has been formalized in the

theory of formal languages. A grammar is ambiguous if there exist two "distinct" derivations of a sentence¹. A language is inherently ambiguous if every grammar generating the language is ambiguous. It has been shown that, 1) there does not exist an algorithm to decide whether a context free grammar is ambiguous; and 2) there does not exist an algorithm to decide whether a context free language is inherently ambiguous [2, 12, 13]. These negative results are important since they prevent research workers from looking for such algorithms. Furthermore, they point to the need of a restricted class of context free languages with the property that the ambiguity problems concerning this class is decidable and this class of languages is powerful enough to specify the "context-free syntax" of the programming languages. The class of deterministic context free languages [14, 17] seems to fit these requirements. However, further research is needed.

8.4 Efficient Parsing Algorithms

An important property of a language is the amount of time necessary to recognize a sentence of the language. A language can be parsed in linear time if there exists an algorithm such that, for an arbitrary input sentence of length n and a specification of the language, the time

¹Let $G = (N, \Sigma, P, S)$ be a grammar. Then, for every sentence x in $L(G)$, there exist a derivation $S \xRightarrow{G} \alpha_1 \xRightarrow{G} \alpha_2 \dots \xRightarrow{G} \alpha_n \xRightarrow{G} x$.

(i. e. number of steps) required by this algorithm to parse this sentence is proportional to n . The general context free grammars, require more than linear time to parse a context free language in general. It has been shown by Younger [46] that there exists an algorithm to recognize context free languages in time n^3 .

When recognizers are really built, context free grammars are rarely used. Restrictions are always put on the grammar to assure that it can be parsed in linear time. The operator precedence grammar defined by Floyd [10] is the most well known one. It is often used to recognize portions of a programming language. Unfortunately, the operator precedence grammars are not rich enough to specify all the "context-free syntax" of the programming languages. Precedence grammars [44], extended precedence grammars [28], simple deterministic context free grammars [22], LR(k) grammars [21] have also been defined. All of them specify classes of languages richer than the class of context free languages. Theoretically they can also be parsed in linear time. However, they are still much less efficient than the operator precedence grammar in a strictly practical sense.

8.5 Conclusion

The body of knowledge in the area of formal languages has been growing very rapidly in recent years. The concept of a grammar has been used for specifying programming languages and building efficient compilers.

The basic drawback in the definition of a formal language is the lack of meaning associated with each sentence. Thus it is not surprising that the application of formal languages appears to be limited to the syntax specification and the parsing algorithms and that various practical problems concerning compilers cannot be answered satisfactorily by the theories in the area of formal languages.

The next step requires the assignment of meaning to each sentence in a formal language. Some investigations indicate that the complexity of the research problem increases considerably but there are some encouraging results. For example, it is known that there does not exist an algorithm to decide whether two arbitrary context free grammars generate the same language [24]. However, it has been shown [35] that there exists an algorithm to decide whether two context free grammars are structurally equivalent in the sense that they not only generate the same language but also "structure" these sentences in the same manner (the same meaning). It is possible that, after assigning "meaning" to each sentence in a language, algorithms to solve some problems which have been proven to be unsolvable at least this seems to be a promising direction for future research.

References

1. Aho, A. V., "Indexed grammars, an extension of context free grammars," Princeton University Doctoral Thesis, 1967.
Also see IEEE Conf. Record of Eighth Annual Symp. on Switching and Automata Theory, Austin, Texas, October, 1967.
2. Cantor, D. G., "On the ambiguity problem of Bacus systems," J. ACM, vol. 9, 1962, pp. 477-479.
3. Chomsky, N., "On certain formal properties of grammars," Inf. Control, 2:2, June, 1959, pp. 137-167.
4. Chomsky, N., "Three models for the description of languages," IEEE-PTIT, 2:3, September, 1956, pp. 113-124.
5. Chomsky, N., "Introduction to the formal analysis of natural languages," Handbook of Math. Psych., vol. 2, Wiley, New York, 1963, pp. 269-321.
6. Chomsky, N. and M. P. Schutzenberger, "The algebraic theory of context free languages," Computer Programming and Formal Systems, North Holland, Amsterdam, 1963, pp. 118-161.
7. Eilenberg, S. and J. B. Wright, "Automata in general algebras," Information and Control, 11:4 (1967), pp. 452-470.
8. Elgot, C. C. and J. E. Mezei, "On relations defined by generalized finite automata," IBM J. Research and Development, January 1965, pp. 47-68.
9. Feldman, J. and D. Gries, "Translator writing system," C. ACM, vol. 11, February 1968, pp. 77-113.
10. Floyd, R. W., "Syntactic analysis and operator precedence," J. ACM, July 1963, pp. 316-353.
11. Floyd, R. W., "The syntax of programming languages—a survey," IEEE-PGEC, 13:4, August 1964, pp. 346-353.
12. Floyd, R. W., "On ambiguity in phrase structure languages," C. ACM, vol. 5 (1962), pp. 526-534.
13. Ginsburg, S. and J. S. Ullian, "Ambiguity in context free languages," J. ACM, vol. 13 (1966), pp. 62-89.

14. Ginsburg, S. and S. A. Greibach, "Deterministic context free languages," Inf. Control, vol. 8 (1966), pp. 620-648.
15. Ginsburg, S., The Mathematical Theory of Context Free Languages, McGraw (1966).
16. Hall, M., The Theory of Groups, The Macmillan Company (1959), pp. 61, 64.
17. Hartmanis, J. P., M. Lewis II, and R. E. Stearns, "Hierarchies of memory limited computations," IEEE Conf. Record on Switching Circuit Theory and Logical Design, Ann Arbor, Michigan, October, 1965, pp. 179-190.
18. Hartmanis, J. and R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Inc. (1966).
19. Held, M. and R. M. Karp, "A dynamic programming approach to sequencing problems," Journal of the Society for Industrial and Applied Mathematics, 10:1, March, 1962, pp. 196-210.
20. Johnson, S. M., "Optimal two- and three-stage production schedules with set-up times included," Naval Research Logistics Quarterly, No. 1 (1954), pp. 61-68.
21. Knuth, D. E., "On the translation of languages from left to right," Inf. Control, vol. 8 (1965), pp. 607-639.
22. Korenjak, A. J. and J. E. Hopcroft, "Simple deterministic languages," IEEE Conf. Record of Seventh Annual Symp. on Switching and Automata Theory, Berkeley, California, October 1966, pp. 36-46.
23. Krohn, K. and J. Rhodes, "Algebraic theory of machines I. Prime decomposition theorem for finite semigroups and machines," Trans. A.M.S., 116:4, April 1965, pp. 450-464.
24. Landweber, P. S., "Decision problems of phrase-structure grammars," IEEE:PGEC, vol. 13 (1964), pp. 354-362.
25. Lawler, E. L., "On scheduling problems with deferral costs," Management Science, 11:2, November, 1964, pp. 280-288.
26. Lawvere, F. W., "Functional semantics of algebraic theories," Proceedings of the National Academy of Sciences, 50:5 (1967), pp. 869-872.

27. Liu, L. Y., "Finite-reversal pushdown automata," Princeton University Doctoral Thesis (1968).
28. Mackeenan, W. M., "An approach to computer language design," Tech. Rept. CS-48, Computer Science Department, Stanford University, Stanford, California, August 1966.
29. MacLane, S., "Categorical algebra," Colloquium Lectures given at Boulder, Colorado, August 27-30, 1963, pp. 26-28.
30. McNaughton, R., "Scheduling with deadlines and loss functions," Management Science, 6:1, October, 1959, pp. 1-12
31. Mezei, J. E. and J. B. Wright, "Algebraic automata and context-free sets," Information and Control, 11:1/2 (1967), pp. 3-29.
32. Moore, J. M., "An n job, one machine sequencing algorithm for minimizing the number of late jobs," to appear.
33. Muller, D. E., Course notes, Mathematics 465, University of Illinois, Fall 1965.
34. Naur, P., "Revised report on the algorithmic language ALGOL 60," C. ACM, vol. 6, January, 1963, pp. 1-7.
35. Paul, M. C. and S. H. Unger, "Structure equivalence of context free grammars," IEEE Conf. Record of 1967 Eighth Annual Symp. on Switching and Automata Theory, Austin, Texas, October 1967.
36. Putzolu, G. R., "Probabilistic aspects of machine structure theory," Doctoral Dissertation, The University of Michigan, January, 1968. Also published as SEL Tech. Rept. no. 26, The University of Michigan, Electrical Engineering Department, December 1967.
37. Putzolu, G. R., "Probabilistic aspects of machine decomposition theory," submitted for publication to the Journal of Computer and System Sciences.
38. Rabin, M. O., "Mathematical theory of automata," in Mathematical aspects of computer science (ed. J. T. Schwartz) Proceedings Symposium on Applied Mathematics, vol. XIX, American Mathematical Society (1967).

39. Rosenkrantz, D. J., "Programmed grammars--a new device for generating formal languages," Columbia University Doctoral Thesis (1967). Also see IEEE Conf. Record of Eighth Annual Symp. on Switching and Automata Theory, Austin, Texas, October 1967.
40. Rothkopf, M. H., "Scheduling independent tasks on parallel processors," Management Science, 12:6, January, 1966, pp. 437-447.
41. Smith, W. F., "Various optimizers for single-stage productions," Naval Research Logistics Quarterly, 3:1/2, March-June, 1956, pp. 59-66.
42. Thatcher, J. W., "A further generalization of finite automata," IBM Research Report RC-1846, June 20, 1967.
43. Whittaker, E. T. and G. N. Watson, A Course of Modern Analysis, Cambridge University Press (1952), p. 237.
44. Wirth, N. and H. Weber, "EULER—a generalization of ALGOL and its formal definition: Part I, Part II," C. ACM, vol. 9, January-February, 1966, pp. 13-25; pp. 89-99.
45. Yeh, R. T., "Weak congruence relations on graphs," Doctoral Dissertation, University of Illinois (1966).
46. Younger, D. H., "Recognition and parsing of context free languages in time n^3 ," Inf. Control, vol. 10, February, 1967, pp. 189-208.
47. Zeiger, H. P., "Cascade synthesis of finite-state machines," Infor. Control, 10, April, 1967, pp. 419-433.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)		
1. ORIGINATING ACTIVITY (Corporate author) University of Michigan Systems Engineering Laboratory Ann Arbor, Michigan 48105		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED
		2b. GROUP N/A
3. REPORT TITLE MATHEMATICAL MODELS OF INFORMATION SYSTEMS		
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Final Report- October 1964 to October 1968		
5. AUTHOR(S) (First name, middle initial, last name) Professor Harvey L. Garner		
6. REPORT DATE August 1969	7a. TOTAL NO. OF PAGES 136	7b. NO. OF REFS 47
8a. CONTRACT OR GRANT NO. AF30(602)-3546	9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO. 5581		
c. No. 558109	9b. OTHER REF. RT NO(S) (Any other numbers that may be assigned this report) RADC TR-69-256	
10. DISTRIBUTION STATEMENT This document has been approved for public release and sale; its distribution is unlimited.		
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Rome Air Development Center (EMIIS) Griffiss Air Force Base, New York 13440
13. ABSTRACT This report summarizes research in the development of mathematical models of information processing systems conducted at The University of Michigan Systems Engineering Laboratory during the period from October, 1967 to October, 1968, under Rome Air Development Center sponsorship. Particular attention is given to a new approach to automata theory, the use of multiple index matrices in generalized automata theory, asymptotic decomposition of machines, recognizability of equation sets, algebraic isomorphism invariants for transition graphs, iterative network realization of sequential machines, optimum sequencing of jobs subject to deadlines, and the theory of formal languages and its impact on the design and implementation of programming languages.		

DD FORM 1473
NOV 68

UNCLASSIFIED

Security Classification

Security Classification

~~UNCLASSIFIED~~
Security Classification